VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Návrh Firewalu v IPv6 sítích

Firewall Design in IPv6 Networks

2019 Tomáš Stiborský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Tomáš Stiborský**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Návrh Firewalu v IPv6 sítích
Firewall Design in IPv6 Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je navrhnout aplikační firewal pomocí Cisco ASA a OS Linux v IPv6 sítích.

Řešení práce spočívá ve splnění následujících bodů:
1. Studium a popis různých typů firewalů.
2. Studium a popis sítí s podporou IPv6 protokolu.
3. Návrh a ověření aplikačního firewalu založeném na Cisco ASA.
4. Návrh a ověření aplikačního firewalu založeném na GNU/Linux.
5. Porovnání a zhodnocení obou návrhů.

Seznam doporučené odborné literatury:

[1] Frahim J., Santos O., Ossipov A. *Cisco ASA: All-in-one Next-Generation Firewall, IPS, and VPN Services.* Cisco Press 2014
[2] Rash M. *Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort.* No Starch Press 2007

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2019

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně. Uvedl všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019 .....

Tímto bych chtěl poděkovat mému vedoucímu práce Ing. Pavlu Nevludovi za cenné rady a předané zkušenosti a všechen čas, který mi věnoval při konzultacích. Také bych rád poděkoval svému kolegovi Ing. Stanislavu Ulmanovi za zapůjčení ASA firewallu, díky kterému mohla být realizována část této diplomové práce. Zároveň bych rád poděkoval své rodině za podporu při studiu.

Abstrakt

V teoretické části diplomová práce popisuje dnešní typy firewallů a obecné přístupy k jejich implementaci, dále popisuje podrobněji principy jejich funkce. Poté práce popsala odlišnosti protokolu IPv6 od protokolu IPv4, hiearchii adresace, typy adres a důsledky v síti plynoucí z těchto odlišností. Poslední teoretická část popisuje protokol 6to4, který je následně otestován i prakticky. Praktická část obsahuje postup implementace firewallu s hlubokou aplikační inspekcí na platformě Cisco – Adaptive Security Apliance a OS Linux. Konkrétním zaměřením aplikačních firewallů jsou protokoly HTTP a FTP, přičemž oba firewally řeší blokaci určitého typu SQL injekce, dále pak blokaci FTP příkazů vzhledem k určenému typu souborů. Práce poskytuje výkonnostní testy obou implementací, srovnání vlivu aplikační inspekce na data přenášená sítí.

Klíčová slova: Linux - firewall, Cisco - Adaptivní bezpečnostní zařízení, hluboká paketová inspekce, IPv6, blokace SQL injekce, aplikační firewall

Abstract

In the theorethical part, the diploma thesis covers current types of firewalls and general ways of their implementation, as well as elaborating on the fundamentals of how they work. Afterwards, the thesis covers differences between IPv6 and IPv4 protocols, address hierarchy, types of addresses and the consequences of those differences. The last section of the theoretical part covers protocol 6to4, which is then tested in practice. The practical part covers the process of implementing firewalls with deep application inspection on platform Cisco – Adaptive Security Appliance and Linux OS. The exact focus of application based firewalls are HTTP and FTP protocols, with both firewalls set to block a certain type of SQL injection attack, as well as blocking FTP commands based on the specified file type. The thesis offers performance tests of both implementations along with comparing the effects of application inspection on the data transferred within the network.

Key Words: Linux - firewall, Cisco - Adaptive security apliance, deep packet inspection, IPv6, SQL injection blocking, application firewall

Obsah

Seznam použitých zkratek a symbolů

ACL	–	Access list
ASA	–	Adaptive Security Apliance
ASIC	–	Application-Specific Integrated Circuit
ASDM	–	Adaptive Security Device Manager
BGP	–	Border Gateway Protocol
CPU	–	Central Processing Unit
DNS	–	Domain Name System
EUI-64	–	64-Bit Extended Unique Identifier
FTP	–	File Transfer Protocol
HE	–	Hurricane Electric
HTTP	–	HyperText Transfer Protocol
HTTPS	–	HyperText Transfer Protocol Secure
ICMP	–	Internet Control Message Protocol
IP	–	Internet Protocol
IoT	–	Internet of Things
IPv4	–	Internet Protocol version 4
IPv6	–	Internet Protocol version 6
LAN	–	Local Area Network
LDAP	–	Lightweight Directory Access Protocol
MAC	–	Media Access Control address
MTU	–	Maximum Transmission Unit
NAT	–	Network address translation
SPI	–	Stateful Packet Inspection
SSH	–	Secure Shell
SQL	–	Structured Query Language
TCP	–	Transmission Control Protocol
TOS	–	Type Of Service
TTL	–	Time To Life
UDP	–	User Datagram Protocol
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
VLAN	–	Virtual Local Area Network
VM	–	Virtual Machine
WAF	–	Web Application Firewall

Seznam obrázků

Seznam tabulek

1 Úvod

Žijeme v době, kdy si většina populace naší planety nedokáže představit život bez internetu a benefitů, které nám tato vymoženost přináší. Dle statistik pro červen 2018 internet na naší planetě aktivně využívá 3,95 miliardy uživatelů. Toto číslo rapidně roste a lidstvo se stává stále závislejší na této technologii. [9] Globálně počet připojených zařízení na světě přesáhl 17 miliard, z čeho je nyní 7 miliard zařízení z odvětví IoT (internet věcí). [10] Těmito fakty se problematika zabezpečení počítačových sítích stává jednou z nejdůležitějších oblastí. Uvážíme-li kolik citlivých informací je dnes v počítačích a na serverech uloženo, požadujeme, aby zabezpečení bylo co největší. V síti IP z pohledu zabezpečení sledujeme tři hlavní cíle:

- Ochranu před neoprávněným přístupem do sítě.
- Ochranu před neoprávněným čtením provozu v síti.
- Zabránit přístupu škodlivého softwaru do sítě.

Přímo úměrně se zvyšováním potřeby internetu a počítačových sítí ke každodennímu životu roste i množství útoků a zneužití těchto technologií proti nám. Je tedy zvýšený důraz na přítomnost firewallu pro každého účastníka moderního internetu. Tato práce porovnává implementaci aplikačního firewallu s použití hardwarového firewallu CISCO Adaptive Security Apliance a alternativní implementaci aplikačního firewallu pomocí GNU Linux. Práce je zaměřená na implementaci obrany v síti s protokolem IPv6, jenž je z důvodu vyčerpání rozsahu IPv4 adres třeba zavádět a rozšiřovat co nejvíce.

2 Firewall

Firewall je zařízení sloužící k oddělení dvou nebo více sítí. Oddělovanými sítěmi můžeme rozumět např. síť LAN a internet, dvě části sítě LAN s různými požadavky na zabezpečení, apod. Jedná se o kontrolní bod, na kterém jsou definována pravidla komunikace – co je povoleno, a co je zakázáno. Existují dvě základní možnosti implementace těchto pravidel.

1. Co není explicitně povoleno, je vždy zakázáno.
2. Nebezpečné služby jsou zakázány, zbytek je povolen.

Nejčastěji používanou a taky doporučenou metodou implementace pravidel je první možnost,jelikož druhá možnost je velice neefektivní a neposkytuje příliš jasný přehled o provozu v síti.

2.1 Rozdělení firewallů podle typu implementace

Firewally můžeme rozdělit mnoha způsoby dle jejich umístění v sítí, licence, principu funkce, atd. Jedno ze základních rozdělení firewally dělí na:

1. Hardwarový firewall
2. Softwarový firewall

Softwarový firewall může být například firewall integrovaný v systému Windows, nebo instalovaný jako přídavný modul antiviru (např.:360totalsecurity Firewall, Avast Firewall). Můžeme taky využít dedikovanou linuxově založenou firewall distribuci jako např: IPFire, nebo využít netfiltr funkce existující v samotném linuxovém jádře.

Hardwarový firewall je fyzické síťové zařízení, které obsahuje modul, jenž dokáže část provozu zpracovávat již v samotném hardwarovém čipu, a tím akceleruje rychlost odbavených paketů skrze firewall. Čip plnící tuto funkci se nazývá ASIC (application-specific integrated circuit). Zástupci v této kategorii jsou například firewally: CISCO ASA, Juniper SRX, Juniper SSG. Toto rozdělení je dnes tak trochu zavádějící a prakticky již nepoužitelné. Trend a princip dnešního firewallu je spíše spojení hardwarového zařízení se softwarem určitého výrobce. Téměř každá z dnes vedoucích značek (Cisco, Juniper, Fortinet, CheckPoint, Palo Alto) poskytuje firewall řešení jako hardwarové zařízení, tak jako software, který může běžet na ESXi serveru ve virtuálním prostředí. Někteří poskytují dokonce verzi implementovatelnou v dnes nejpoužívanějších cloud platformách (Amazon Web Services, Azure, Google Cloud Platform).

2.2 Rozdělení firewallu podle sítových vrstev

Při klasifikaci firewallů musíme zahrnout také rozdělení z pohledu zpracování provozu. Dle tohoto kritéria můžeme firewally rozdělit do tří velkých skupin:

1. Paketový filtr
2. Stavový firewall
3. Aplikační brána

2.2.1 Paketový filtr

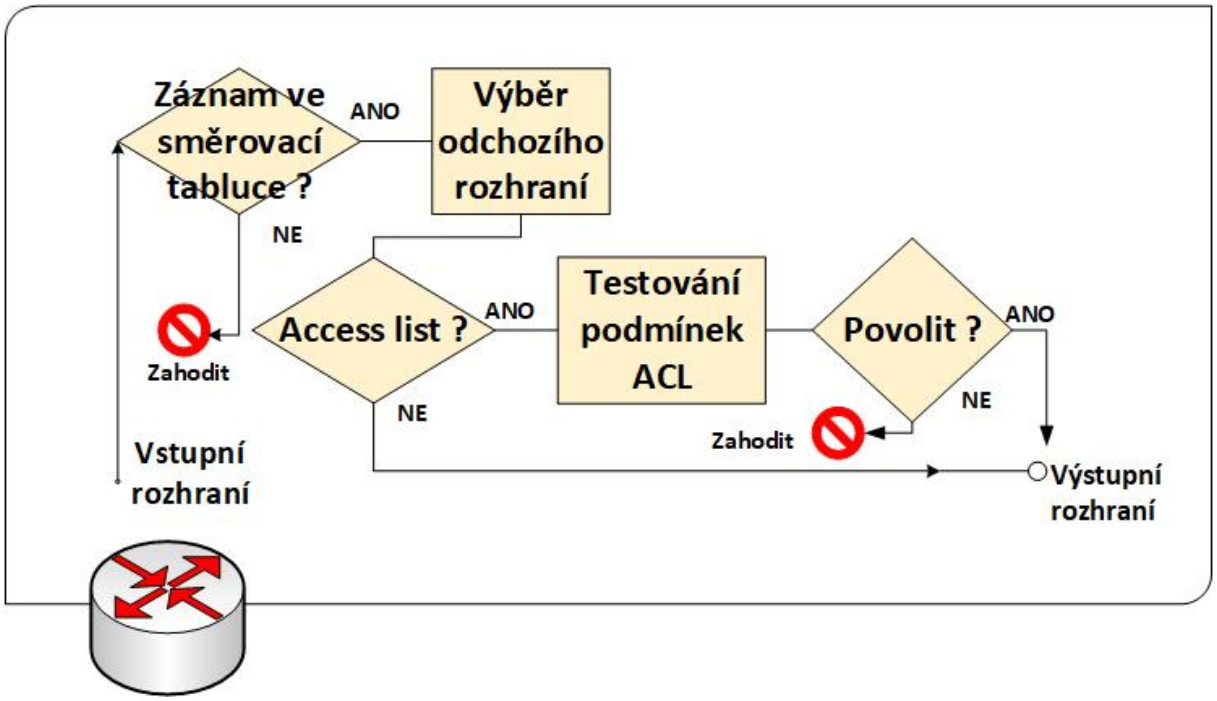
Je svým principem nejjednodušším typem firewallu. Pracuje na třetí nebo čtvrté síťové vrstvě a jeho podstata spočívá v definici pravidel přístupu (z ang: Access list (ACL)). Pravidla definují, která IP smí komunikovat s jinou IP, a na jakém portu smí, či nesmí komunikovat. Výhodou tohoto firewallu je jeho systémová nenáročnost. Můžeme ho implementovat třeba i na směrovači. Pokud například chceme povolit PING, SSH, a HTTP z IPv6 internetu na zařízení 2001:470:736e:c::1 musíme definovat pravidla pro příchozí směr viz.: Tabulka:1 a poté definovat druhý seznam pravidel pro opačný směr, aby mohlo zařízení odpovídat viz.: Tabulka:2. Vnitřní algoritmus použití paketového filtru na směrovači je znázorněn pomocí diagramu viz.: Obrázek 1.

Pořadí	Akce	Protokol	Zdrojová IP	Zdrojový Port	Cílová IP	Cílový port
1	povolit	icmp6	any6	*	2001:470:736e:c::1	
2	povolit	tcp	any6	*	2001:470:736e:c::1	80
3	povolit	tcp	any6	*	2001:470:736e:c::1	22
4	zakázat	IP				

Tabulka 1: Tabulka pravidel pro příchozí směr provozu.

Pořadí	Akce	Protokol	Zdrojová IP	Zdrojový Port	Cílová IP	Cílový port
1	povolit	icmp6	2001:470:736e:c::1	*	any6	
2	povolit	tcp	2001:470:736e:c::1	80	any6	*
3	povolit	tcp	2001:470:736e:c::1	22	any6	*
4	zakázat	IP				

Tabulka 2: Tabulka pravidel pro odchozí směr provozu.



Obrázek 1: Paketový filtr na směrovači.

2.2.2 Stavový firewall

Též známý jako stavový paketový filtr (anglicky stateful firewall) je v informatice označení pro takový firewall, který podporuje SPI (anglicky Stateful packet inspection), což znamená, že je schopen sledovat a udržovat všechny navázané TCP/UDP relace (pracuje na transportní vrstvě referenčního modelu ISO/OSI). Stavový firewall je schopen rozlišovat různé stavy paketů v rámci jednotlivých relací (spojení) a jeho úkolem je propustit pouze takové, které patří do již povolené relace (jiné jsou zamítnuty)[18].

Stavový firewall sleduje stav síťových připojení (například streamy TCP nebo komunikaci UDP) a je schopen udržovat významné atributy každého spojení v paměti. Tyto atributy jsou kolektivně označovány jako stav připojení a mohou obsahovat takové podrobnosti jako IP adresy, zdrojové a cílové porty jednotlivých spojení a pořadová čísla paketů procházejících spojením. Stavová kontrola sleduje příchozí a odchozí pakety v průběhu času, stejně jako stav připojení a ukládá data do dynamických stavových tabulek. Tato data jsou vyhodnocována tak, aby rozhodnutí o filtrování nebyla založena pouze na pravidlech definovaných správcem, ale také na kontextu, který byl vytvořen předchozími připojeními.

Nejvíce intenzivní zatížení pro CPU je kontrola, která se provádí v době nastavení připojení. Položky ve stavové tabulce jsou vytvářeny pouze pro TCP nebo UDP spojení, které splňují definované bezpečnostní zásady. Pak jsou všechny pakety (pro danou relaci) zpracovány rychle, protože je jednoduché a rychlé určit, zda patří k existující sestavené relaci. Pakety sdružené s těmito relacemi mohou procházet firewallem. Relace, které neodpovídají žádnému pravidlu, jsou

zamítnuty jako pakety, které neodpovídají existující položce stavové tabulky.

Chcete-li zabránit tomu, aby se tabulka stavů naplnila, využívá se mechanismus stárnutí spojení. Pokud po určitou dobu neproběhla žádná komunikace (typicky 300 s). Dané spojení zestárne a tato spojení jsou odstraněna ze stavové tabulky. Mnoho aplikací proto pravidelně odesílá zprávy pro udržování spojení, aby firewall zastavil proces stárnutí připojení, a tím pádem předchází zahazování spojení firewallem během období bez aktivity uživatele.

V závislosti na protokolu připojení je udržování stavu připojení více či méně složité pro firewall. Například protokol TCP je ve své podstatě protokolem, jehož spojení jsou sestaveny trojcestným navázáním („SYN, SYN-ACK, ACK“) a ukončeny výměnou „FIN, FIN-ACK, ACK“. To znamená, že všechny pakety s „SYN“ v hlavičce přijaté firewallem jsou interpretovány tak, že otevírají nová připojení. Pokud je služba požadovaná klientem k dispozici na serveru, bude reagovat paketem „SYN-ACK“, který bude firewall také sledovat. Jakmile brána firewall obdrží odpověď klienta „ACK“, převede připojení do stavu „ESTABLISHED“, protože spojení bylo ověřeno obousměrně. To umožňuje sledování budoucích paketů prostřednictvím zavedeného připojení.

Jiné protokoly připojení, jmenovitě protokol UDP a protokol ICMP, nejsou založeny na obousměrném potvrzení jako je protokol TCP, což způsobuje, že stavový firewall je o něco méně bezpečný. Chcete-li sledovat stav připojení v těchto případech, brána firewall musí převést relace do stavu „ESTABLISHED“ ihned poté, co příjme první platný paket. Pak může sledovat spojení pouze prostřednictvím adres a portů zdroje a cíle v paketu. Na rozdíl od protokolu TCP, kde mohou být spojení uzavřeny výměnou „FIN, ACK“, tyto bezspojové protokoly umožňují ukončení relace pouze časovým limitem.

Sledováním stavu připojení poskytují stavové firewally větší efektivitu z hlediska kontroly paketů. Je to proto, že u existujících připojení musí brána firewall zkontrolovat pouze stavovou tabulku namísto kontroly paketu proti sadě pravidel brány firewall, která může být rozsáhlá. Navíc v případě shody se stavovou tabulkou brána firewall nepotřebuje provádět hloubkovou inspekci paketů. [18]

2.2.3 Aplikační brána

Tento druh firewallu vylepšuje vlastnosti stavového firewallu o funkci hloubkové inspekce na sedmé vrstvě ISO/OSI. Kontroluje tedy nejen port, se kterým je příslušná služba spjatá, ale dokáže kontrolovat i příslušný obsah aplikačního protokolu. Dle principu jakým firewall zpracovává provoz můžeme tento typ firewallu dále rozdělit na dvě podskupiny.

- Proxy firewall
- Firewall nové generace

Proxy firewall Základem tohoto řešení je předpoklad, že uživatel kontaktuje předem známou službu běžící na serveru (FTP, HTTP, HTTPS). Spojení ale není navázáno přímo na server

s poskytovanou službou, ale nejdříve na virtuální adresu reprezentující danou službu vytvořenou bránou proxy. Brána ověří uživatele, analyzuje dotaz, porovná s databází známých útoků a pokud splňuje povolená specifika, naváže spojení k cílovému serveru. Proxy svým fungováním plní funkci NAT a cílové servery se nikdy nedozví reálnou zdrojovou adresu příchozího požadavku. Příchozí provoz z pohledu serveru je vždy vnitřní adresa proxy brány. Jelikož komunikace probíhá skrze dvě separátní spojení, může brána zasahovat do samotných aplikačních dat. Typickou podskupinou implementace, která tuto vlastnost využívá, je:

Web Application Firewall(WAF), jenž je proxy firewall zaměřený na webové aplikace, jehož primárním cílem je preventivně zahazovat spojení obsahující známky známých útoků proti webovým službám (SQL injection, Cross Site Scripting (XSS)). Příklad scénáře použití WAF firewallu se zásahem do aplikačních dat: Virtuální adresa proxy naslouchá na protokolu HTTPS a vlastní certifikát a privátní klíč pro danou doménu. Uživatel se připojuje ověřeným šifrovaným spojením k této virtuální adrese. Proxy spojení dešifruje a provede požadovanou kontrolu, pokud je spojení bezpečné, naváže druhé spojení k serveru s příslušnou webovou aplikací pomocí nešifrovaného spojení (HTTP). Tento systém tak přenáší zátěž úkonu dešifrování z koncových serverů do centrálního bodu a poskytuje komplexnější přehled o použitých certifikátech a data jejich expirace. V případě objevené zranitelnosti, či potřeby přidané bezpečnosti pomocí HTTPS, k již napsané HTTP aplikaci, není třeba měnit nic na koncové aplikaci/serveru, ale o vše se postará právě WAF. Tato vlastnost je v korporátní sféře velice žádaná. [3]

WAF můžeme implementovat několika způsoby:

• **Reverzní proxy**, při této implementaci je aplikační brána logicky umístěna mezi klienty a webovým serverem. Veškeré požadavky přichází nejdříve na proxy bránu, kde jsou filtrovány. Pouze povolené požadavky jsou poté přes vnitřní infrastrukturu sítě směrovány k danému serveru. Výhoda této implementace je, že se nebezpečný a neověřený požadavek nikdy nedostane na cílový server.

• **Most 2. vrstvy**, tato implementace je svým přístupem k provozu stejná jako reverzní proxy s rozdílem, že se provoz přeposílá pouze na 2. vrstvě ISO/OSI modelu. Implementace L2 mostu je oproti reverzní proxy jednodušší, funkce bývá rychlejší, ale neumožňuje tak rozsáhlou bezpečnostní politiku (např.: nemůžeme dešifrovat provoz a změnit na HTTP).

• **„Out-of-band“ řešení**, při řešení WAF jako „out-of-band“ se využívá funkce zrcadlení provozu na L2 přepínači. Provoz z rozhraní k serveru zrcadlíme na rozhraní směrem k firewallu. Nevýhodou řešení je fakt, že blokace probíhá pouze promocí přerušení spojení pomocí TCP-RST. Firewall nemůže nějak zasahovat do provozu a provádět složitější bezpečnostní politiku. Proto se toto řešení používá pouze k monitorování webového provozu pro daný server.

• **Implementace na serveru**, tento typ WAF je implementován jako dodatečná aplikace na webovém serveru. Aplikace firewallu ale využívá serverové zdroje (CPU,RAM,HDD), a tím pádem mohou chybět webové aplikaci, kterou má firewall chránit. Z tohoto důvodu není implementace na serveru možná použít všude, a při jejím použití je třeba počítat s dostatečně velkou hardwarovou rezervou. [3]

• **Cloudové řešení**, poslední technika implementace přesměruje ochranu na externího dodavatele. Prakticky se jedná o systém s reverzní proxy s tím rozdílem, že proxy není umístěna v námi spravované infrastruktuře, ale v síti externí společnosti. Do naší infrastruktury přichází bezpečné a filtrované požadavky již na úrovni internetu (filtrace probíhá v cloudu).

Firewall nové generace Čím dál větší podíl provozu v síti je dnes šifrován. Šifrování je prvek zvyšující bezpečnost, ale může být také zneužit jako krycí mechanismus moderního útoku. Klasické firewally jsou proti těmto útokům bezradné. Možností, jak tento problém řešit, je poslední stupeň vývoje aplikační brány – Firewall nové generace (Next generation firewall).

Firewally nové generace poskytují veškeré funkce předchozích firewallů, ale navíc přinášejí funkci integrované dekrypce a hluboké inspekce aplikačních dat. Jsou tedy schopny kontrolovat provoz nejen na základě adres a portů, ale obsahují aktualizované databáze přesného chování jednotlivých aplikací a definice jejich zranitelností. Jakmile je provoz firewallem dešifrován, aplikace jako facebook chat je rozpoznána a kontrolována. Dekryptovaná data mohou být inspektováná proti hrozbám, může být aplikováno URL filtrování, blokování přenosu souborů, nebo blokování nebezpečných dat [11],[12].

Oproti zneužití slabých stránek síťových komponent a služeb, je více než 80% všech nových malware a pokusů o vniknutí využívá slabé stránky v aplikacích.

Stavové firewally s jednoduchými funkcemi filtrování paketů účinně blokovaly nežádoucí aplikace, protože většina aplikací splňovala očekávání portu a protokolu. Administrátoři mohli okamžitě zabránit tomu, aby uživatelé mohli přistupovat k nebezpečné aplikaci blokováním přidružených portů. Dnes však blokování webové aplikace jako je Farmville, která používá port 80 pomocí uzavřením portu, by také znamenalo komplikace s celým HTTP protokolem. [12]

Mezi všemi výrobci firewallů nové generace byl vyjednán obecný seznam vlastností, který musí firewall splňovat, aby byl považován za firewall nové generace. [17]

- Aplikační povědomí (Application Awareness) – Definice chování aplikací a jejich zranitelností.
- Stavová inspekce (Stateful Inspection) – viz.: 2.2.2
- Integrovaný systém ochrany proti vniknutí (IPS)
- Detekce identity (Uživatelská a skupinová kontrola)
- Možnost módu směrování, módu mostu
- Možnost využití externích zdrojů správy – kontrola databází LDAP / Active Directory

3 IPv6 – Internet protokol verze 6

Základním kamenem IPv6 je dokument RFC 2460: Internet Protocol, Version 6 (IPv6) Specification, který popisuje formát datagramu. Všem ostatním mechanismům, které souvisejí s IPv6, jsou věnovány další RFC specifikace. [1]

3.1 Datagram IPv6

Formát datagramu IPv6 je podobného tvaru jako IPv4. Datagram začíná hlavičkami, za které se připojují data. Dříve byla délka hlaviček proměnlivá, jelikož jednotliví účastníci komunikace mohli přidávat další nepovinné volby podle potřeby. V IPv4 hlavička obsahuje kontrolní součet, který se musí znovu vypočítat na každém směrovači, jelikož průchod sítí snižuje hodnotu TTL. [1]

V IPv6 je standardní hlavička minimalizována a její prvky byly omezeny jen na ty nejnutnější. Základní hlavička má tedy konstantní velikost. Veškeré doplňující, či méně používané údaje byly přesunuty do rozšiřujících hlaviček, které v datagramu mohou, a nemusí být přítomny.[1]

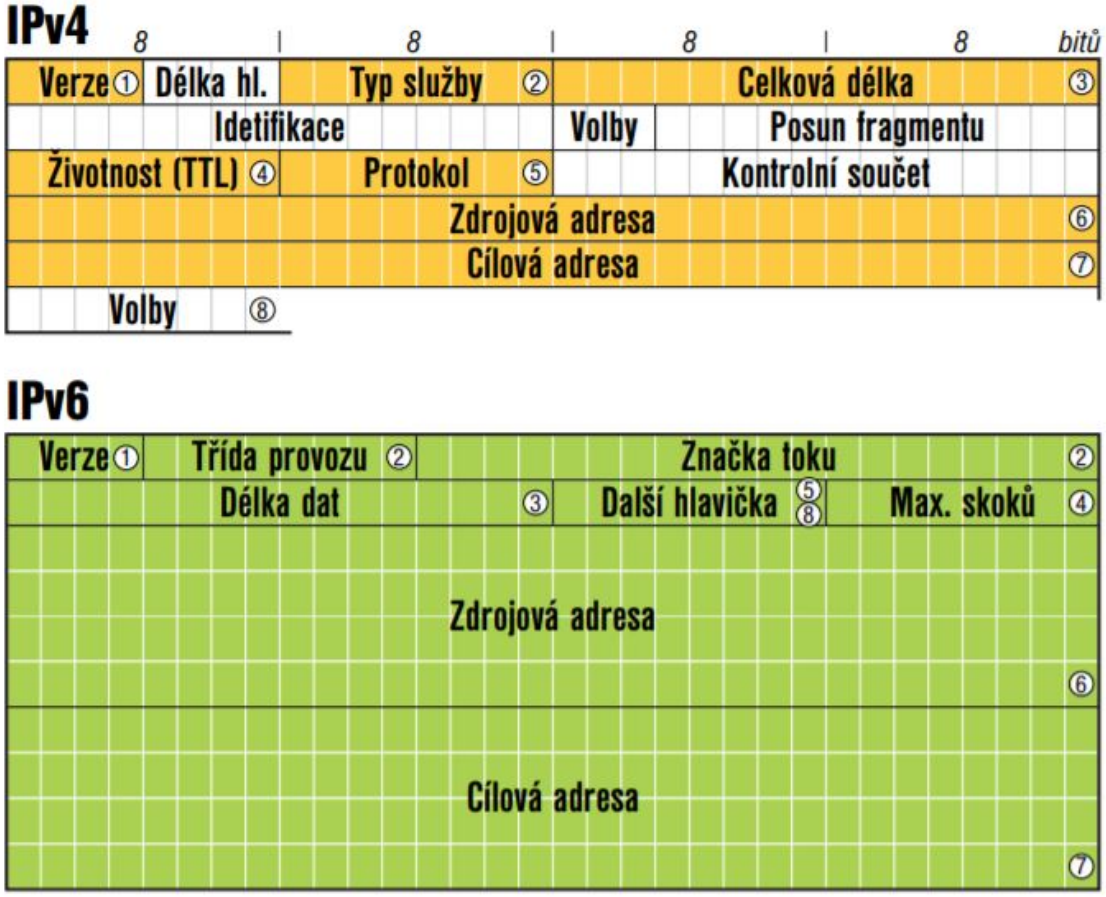
Porovnání hlaviček obou protokolů vidíte na Obrázku: 2. Je vidno, že se adresy odesilatele a příjemce prodloužily čtyřikrát, ale celková délka základní hlavičky datagramu vzrostla ve srovnání s IPv4 jen dvojnásobně (z 20 B na 40 B, z toho 32 B zabírají adresy). Minimalismus je patrný na první pohled.[1]

Verze Položka Verze (Version) je obvyklým zahájením IP datagramu, má úkol identifikovat verzi protokolu. V IPv6 obsahuje hodnotu 6. [1]

Třída provozu Za ní následuje osmibitová Třída provozu (Traffic class), která vyjadřuje prioritu datagramu či jeho zařazení do určité přepravní třídy. Cílem je, aby tato položka umožnila IP poskytovat služby se zaručenou kvalitou. V praxi ale IP protokol, a to ani ve verzi 6, neumí zaručit dopravní parametry jako jsou přenosová rychlost, zpoždění, či jeho rozptyl. Dovede však poskytovat takzvané diferencované služby (differentiated services, diffserv). Jejich prostřednictvím můžeme datagramům přiřadit různé priority a odlišné způsoby zacházení, které vedou k jejich prioritnímu zpracování, či naopak odkládání až po ostatních. Právě diferencované služby využívají pro prioritizaci provozu informaci obsaženou v poli Třída provozu. [1]

Značka toku Dalších 20 bitů je věnováno Značce toku (Flow label). Koncepce toku je novinkou v IPv6 a stejně jako třída provozu zatím není přesně definována. Záměrem je identifikovat tok jako proud datagramů se společnými vlastnostmi (odesilatel, adresát, požadavky na vlastnosti spojení). Prostřednictvím identifikátoru (značky) směrovač rychle rozpozná, že datagram je součástí určitého toku, což mu usnadní rozhodování o jeho dalším osudu. Jedná se stále o experimentální půdu a nic konkrétního zatím nebylo stanoveno.[1]

Délka dat Délka dat (Payload length) nese informaci o délce datagramu. Přesněji řečeno počet bajtů následujících za standardní hlavičkou. Základní hlavička se do této délky nepočítá, zatímco rozšiřující hlavičky ano. Jelikož je položka dvoubajtová, je maximální délkou 64 KB. Pokud je třeba vytvořit delší datagram, lze použít rozšiřující hlavičku Jumbo.[1]



Obrázek 2: Porovnání hlaviček IPv4 proti IPv6, přejato z:[1, str.37].

Další hlavička (Next header) obsahuje identifikaci, hlavičky či druhy dat následujících za standardní hlavičkou. [1]

Dosah Maximální počet skoků (Hop limit) je náhrada dřívější životnosti datagramu (TTL). Průchod datagramu jedním směrovačem je považován za jeden skok. Smyslem omezení skoků je ochrana proti cyklům při směrování. [1]

Adresy Závěrečnými dvěma položkami je dvojice IPv6 adres: Zdrojová adresa (Source address) a Cílová adresa (Destination address). Vzhledem k délce adresy v IPv6 zabírají tyto dvě položky 80 % rozsahu celé hlavičky. [1]

Při srovnání IPv6 s IPv4 je nejnápadnější absence tří informací: rozšiřující volby, kontrolní součet a fragmentace. Rozšiřující volby byly nahrazeny obecnějším principem zřetězení doplňkových hlaviček. Údaje o fragmentaci byly přesunuty do těchto rozšiřujících hlaviček. Zdaleka ne každý paket je totiž fragmentován a lze očekávat, že v IPv6 bude fragmentace ještě vzácnější než v současnosti. IPv6 totiž požaduje, aby infrastruktura pro jeho přenos dovedla přenášet pakety minimálně o délce 1280 B (MTU). Vzhledem k tomu, že drtivá většina koncových zařízení je dnes připojena prostřednictvím různých variant Ethernetu s MTU 1500 B, lze očekávat, že se

tato velikost paketů usídlí všude a fragmentace zmizí ze světa sítí. [1]

Kontrolní součet zmizel bez náhrady. Tuto službu typicky vykonává nižší vrstva síťové architektury (např. zmiňovaný Ethernet). Vzhledem k tomu, že hlavička se mění v každém směrovači (klesá dosah datagramu), znamenalo by to zbytečné zpomalování.[1]

3.2 Adresace v IPv6

V IPv6 – stejně jako u jeho předchůdce – jsou adresy přiřazovány síťovým rozhraním, nikoli počítačům. Má-li váš počítač dvě síťové karty, bude mít každá z nich svou IP adresu.

Odlišujeme tři druhy adresy podle jejich chování.

- Individuální (unicast) Každá z nich identifikuje jedno síťové rozhraní a data mají být dopravena pouze jemu.
- Skupinové (multicast) slouží pro adresování skupin počítačů či jiných zařízení v síti. Pokud někdo odešle data na tuto adresu, musí být doručena všem členům skupiny.
- Výběrové (anycast) představují novinku v IPv6. Také výběrové adresy označují skupinu, data se však doručí pouze jedinému jejímu členovi – tomu, který je nejblíže. [1]

Porovnání s IPv4 zmizely všesměrové (broadcast) adresy. Nejsou potřeba, protože jejich funkce přebírají adresy skupinové. IPv6 definuje speciální skupiny, např. pro všechny uzly na dané lince, které umožňují plošnou distribuci zpráv. [1]

• **Zápis adresy** Standardním způsobem zápisu IPv6 adresy je osm skupin po čtyřech číslicích šestnáctkové soustavy, které vyjadřují hodnoty 16 bitů dlouhých částí adresy. Navzájem se oddělují dvojtečkami.[1] Příkladem IPv6 adresy je:

fedc:ba98:7654:3210:fedc:ba98:7654:3210

• **Zkracování** Jelikož je poměrně častou hodnotou nula, nabízí se dvě možnosti pro zkrácení zápisu. V každé čtveřici můžete vynechat počáteční nuly. Místo „0000“ tedy lze psát jen „0“. Někdy se dokonce vyskytuje několik nulových skupin za sebou. Ty můžete nahradit zápisem „::“. Například adresu:

0123:0000:0000:0000:fadc:bc98:76c4:3c10

můžeme zkrátit na:

123:0:0:0:fadc:bc98:76c4:3c10

nebo dokonce jen na:

123::fadc:bc98:76c4:3c10

Koncovou nulu (v poslední čtveřici) pochopitelně vynechat nelze. Kdybyste napsali jen „321“, znamenalo by to „0321“, nikoli „3210“. Úplný extrém představuje nedefinovaná adresa:

0000:0000:0000:0000:0000:0000:0000:0000

kterou lze zkrátit až na samotné „::“

Konstrukci „::“ můžete v každé adrese použít jen jednou. Jinak by nebylo jednoznačné, jak se má adresa rozvinout do původní podoby. [1] Například adresu:

0123:0000:0000:0000:4567:0000:0000:0000

můžete psát jako

123::4567:0:0:0 nebo 123:0:0:0:4567::
nikoli však: 123::4567::

3.3 Typy adres

Obrovský adresní prostor, který má IPv6 k dispozici, byl rozdělen do několika skupin – typů adres. Každý typ sdružuje adresy se společnou charakteristikou. Příslušnost k jednotlivým typům určuje prefix adresy.[1]

Prefix	Význam
::/128	nedefinovaná adresa
::1/128	smyčka (loopback)
fc00::/7	unikátní individuální lokální
fe80::/10	individuální lokální linkové
ff00::/8	skupinové adresy
ostatní	individuální globální
	známé prefixy
64:ff9b::/96	adresy s vloženým IPv4
2001::/32	Teredo
2001:db8::/32	adresy pro příklady v dokumentech
2002::/16	6to4

Tabulka 3: Základní rozvržení adres a prefixů.

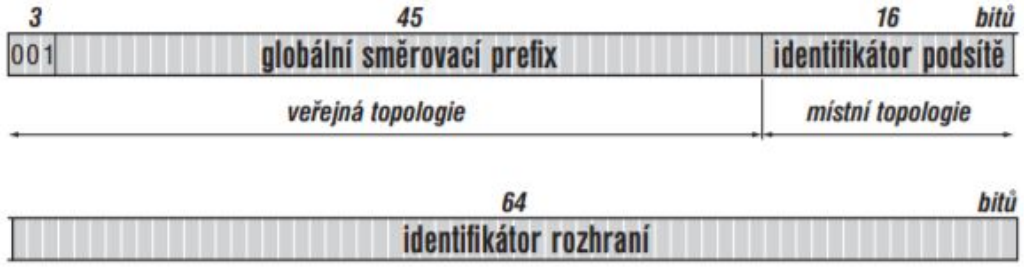
3.3.1 Globalní inviduální adresy

Slovo globální naznačuje, že identifikují svého nositele v rámci celého Internetu a musí tudíž být celosvětově jednoznačná. Zatím byla definována jen část z nich (prefix 2000::/3), které se nacházejí v RFC 3587: IPv6 Global Unicast Address Format. [1]

Globální adresy jsou přidělovány hierarchicky podle pravidel podobných CIDR ze světa IPv4. To znamená, že poskytovatel Internetu (neboli lokální registr, LIR) obdrží určitý prefix, jehož části v podobě delších prefixů se shodným začátkem pak přiděluje svým zákazníkům. Cílem tohoto přístupu je agregace směrovacích údajů – aby bylo možné, celou poskytovatelovu síť se všemi jeho zákazníky popsat jediným záznamem ve směrovací tabulce, právě tímto společným

prefixem. Tato agregace je velice důležitá, protože významným způsobem zmenšuje velikost směrovacích tabulek. [1]

RFC 3587 zavedlo maximálně zjednodušený model přístupu ke globálnímu adresnímu prostoru, v podstatě odpovídající struktuře adresy pro IPv4. Ta má tři části: adresu sítě, podsítě a rozhraní v podsíti. Analogické části má i IPv6 adresa, kde adresa sítě byla přejmenována na globální směrovací prefix. Jejich délky jsou definovány zcela obecně, avšak podle současných pravidel přidělování globální směrovací prefix měří nejčastěji 48 bitů, adresa podsítě 16 bitů a adresa rozhraní v podsíti 64 bitů.[1] Strukturu globální individuální adresy s nejobvyklejšími délkami jednotlivých částí znázorňuje Obrázek: 3.



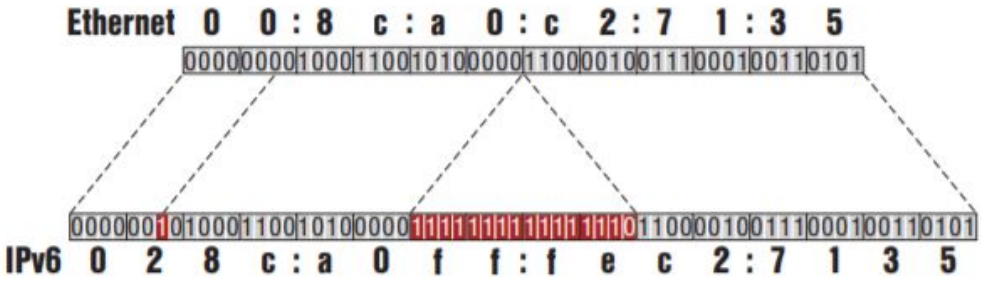
Obrázek 3: Struktura globální individuální adresy, přejato z: [1, str.60].

Identifikátory rozhraní – modifikované EUI-64 Základní podoba identifikátoru rozhraní v IPv6 je odvozena z IEEE EUI-64. Jedná se o standard zaměřený na přidělování globálních identifikátorů pro rozhraní v počítačových sítích. Jejich délka je 64 bitů a odpovídá tedy délce vyhrazené pro rozhraní v IPv6 adrese. [1]

Nejjednodušším případem je, když dané rozhraní má již přidělený EUI-64 identifikátor. Ten se do IPv6 adresy převezme s jednou změnou. Předposlední (druhý nejméně významný) bit v nejvyšším bajtu EUI-64 identifikátoru slouží jako příznak globality. Ve standardním EUI-64 zde hodnota 0 signalizuje celosvětově jednoznačnou adresu, zatímco 1 označuje adresu lokální. IPv6 používá modifikované EUI-64 a hodnotu tohoto bitu invertuje. Čili 0 představuje lokální identifikátor a hodnota 1 identifikátor globální. [1]

Nejčastějším případem jsou sítě založené na některé z variant Ethernetu či bezdrátové sítě IEEE 802.11. V takovém případě mají jednotlivá rozhraní výrobcem přidělené celosvětově jednoznačné 48 bitové MAC adresy. Jejich transformace na modifikované EUI-64 je velmi snadná a standardní: mezi třetí a čtvrtý bajt MAC adresy se vloží 16 bitů s hodnotou fffe. Mimo jiné se také obrátí příznak globality. Například z MAC adresy 00:8c:a0:c2:71:35 se v IPv6 adrese stane identifikátor rozhraní 028c:a0ff:fec2:7135.

Používání EUI-64 je velmi přímočaré, proto se proti němu ozývají ochránci soukromí uživatelů. Jelikož je EUI-64 odvozeno z MAC adresy, která je celosvětově jednoznačná a mění se jedině s výměnou síťové karty, vzniká tak jednoznačný identifikátor počítače. Navíc tato část adresy zůstává stejná, i když se počítač pohybuje, protože počítač si svou adresu generuje stále stejně:



Obrázek 4: Modifikovaný EUI-64 z ethernetové adresy, přejato z:[1, str.62].

zjistí prefix zdejší podsítě a připojí k němu svůj identifikátor rozhraní. Pokud by se se někomu podařilo odposlouchávat síťový provoz na strategických místech, mohl by sledovat, s kým vším daný počítač komunikuje a jak se pohybuje po světě. Nepomůže proti tomu ani šifrování, protože se šifruje pouze obsah datagramu. Adresy musí zůstat otevřené. Reakcí na tyto problémy je RFC 4941: Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC v podstatě navrhuje náhodné generování identifikátorů rozhraní, které mohou mít životnost několik hodin až dnů a počítač je může neustále měnit. Ovšem je potřeba mít i nějaký pevný bod, aby se s počítačem dalo vůbec navázat spojení. Proto RFC 4941 navrhuje, aby počítač měl jeden pevný identifikátor rozhraní (podle EUI-64), pod nímž bude zaveden v DNS. Hlavním smyslem této adresy je její použití jako cílový bod pro komunikaci navazovanou zvenčí. Dle RFC si počítač generuje náhodné dočasné identifikátory. Adresám z nich odvozených dává přednost pokud sám navazuje spojení s někým jiným. Díky tomu klienti na daném počítači používají náhodné krátkodobé adresy a nelze dlouhodobě sledovat jejich aktivity.

RFC 3972 zavedlo další odrůdu identifikátorů rozhraní, jež umožňují zabezpečit objevování sousedů. Vycházejí z veřejného klíče svého vlastníka a znemožňují nepřátelské stanici vydávat se za někoho jiného. [1]

3.3.2 Lokální adresy

Koncept adres, které neplatí v celém Internetu, ale pouze v jeho malé části, zavedlo RFC 1918: Address Allocation for Private Internets. Malou část adresního prostoru IPv4 vyhradilo pro neveřejné adresy, které lze používat v koncových sítích, ale nejsou směrovány za jejich hranicemi. Tyto adresy nejsou celosvětově jednoznačné, každá koncová síť si s nimi může nakládat, jak se jí zlíbí. Původně byly určeny především pro experimenty či pro sítě, které neměly ambice připojit se k Internetu. V současnosti se používají masově v kombinaci s NATem, který jim přístup k Internetu dokáže zprostředkovat.[1]

IPv6 zavádí koncept dosahu adres. Ten je ale přínosný hlavně pro skupinové adresy, jejichž součástí je přímo informace o dosahu. Pro individuální adresy jsou možnosti omezenější, nicméně i zde existuje několik typů adres s omezeným dosahem.[1]

• **Lokální linkové** Největší význam mají lokální linkové adresy (link local). V adresní architektuře mají svou vyhrazenou část s prefixem fe80::/10. Následujících 54 bitů je nulových, za nimi se nachází 64bitový identifikátor rozhraní podle modifikovaného EUI-64. Jejich hlavní výhodou je, že počítač si takovou adresu dokáže vygenerovat sám a nepotřebuje k tomu žádnou infrastrukturu. Díky tomu je lokální linková adresa k dispozici vždy. Tudíž stačí propojit počítače ethernetovým kabelem, není potřeba mít žádný směrovač ani DHCP server, a přesto mohou rovnou komunikovat prostřednictvím lokálních linkových adres, které si samy vytvoří.[1]

• **Lokální místní** Roli velmi podobnou adresám z RFC 1918 hrály ve starších definicích lokální místní adresy (site local). Byl jim přidělen prefix fec0::/10 a jejich platnost byla omezena na jedno „místo“. Typickým místem je koncová síť organizace připojené k Internetu. Jenže existují také organizace připojené k Internetu v několika lokalitách téhož města, či dokonce v různých městech a státech. Mají být areály firmy v Ostravě, v Opavě, v Bratislavě a v Brně považovány za čtyři různá místa, nebo za jedno místo? Praxe ukázala, že definice místa je vágní a její výklad se pro každého velmi liší. Navíc se ukázaly problémy s konfiguracemi směrovačů a další obtíže při pokusech o reálné použití místních adres.[1]

Výsledkem bylo RFC 3879: Deprecating Site Local Addresses, které místní lokální adresy zamítlo a dokonce zakazuje novým implementacím podporovat zpracování adres s prefixem fec0::/10.

• **Unikátní lokální** Jejich nástupcem se staly unikátní lokální adresy (unique local, ULA) definované v RFC 4193: Unique Local IPv6 Unicast Addresses. Poznají se podle prefixu fc00::/7. Za ním následuje jednobitový příznak L, zda byl prefix adresy přiřazen lokálně (L=1) nebo jinak. Vzhledem k tomu, že všechny v současnosti používané adresy tohoto typu jsou generovány lokálně, mají nastaven příznak L na hodnotu 1 a začínají proto vždy prefixem fd00::/8. Dalších 40 bitů obsahuje globální identifikátor, kterým je náhodně vygenerované číslo. Za ním následuje v adrese vše podle standardního postupu: 16bitový identifikátor podsítě a 64bitový identifikátor rozhraní podle modifikovaného EUI-64.[1]

3.3.3 Adresy obsahující IPv4

Některé přechodové mechanismy potřebují vyjádřit adresy, které pocházejí ze sítě IPv4. Aktuálně se k tomuto účelu používá formát zavedený v RFC 6052: IPv6 Addressing of IPv4/IPv6 Translators a dané adresy s vloženým IPv4 (IPv4-embedded). Využívají faktu, že adresní prostor IPv4 je mnohem menší, proto lze vyčlenit část IPv6 prostoru a použít ji pro reprezentaci IPv4. Tato část je identifikována určitým prefixem a mapovaná adresa vznikne připojením IPv4 adresy za prefix.[1]

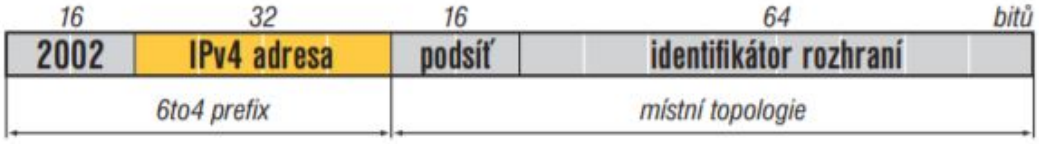
Prefix může být dvou typů – může se jednat o prefix, který přidělí správce z vlastního adresního prostoru. Vzhledem ke standardní interpretaci bitů v identifikátoru rozhraní musí bity číslo 64 až 71 obsahovat nulových hodnot. Je doporučeno vytvořit prefix tak, že se vyjde z prefixu délky 64 b a doplní se na požadovanou délku nulami. Druhou variantou bude použití univerzálního prefixu 64:ff9b::/96 definovaného pro tyto účely přímo v RFC 6052. Pouze při jeho

použití se může vložená IPv4 adresa zapsat ve standardním tvaru – v desítkové soustavě s bajty oddělenými tečkami. Adresu s univerzálním prefixem obsahující 147.230.1.2 lze tedy zapsat ve tvaru 64:ff9b::147.230.1.2 nebo 64:ff9b::93e6:102. Zatímco při mapování pomocí vlastního prefixu je přípustný jen druhý tvar, například 2001:db8:ff:ee::93e6:102. [1]

3.4 6to4

Hlavním cílem 6to4 je umožnit koncovým IPv6 sítím vzájemnou komunikaci procházející IPv4 Internetem a to s minimální konfigurací. Požaduje jen, aby připojovaná síť měla k dispozici alespoň jednu veřejnou IPv4 adresu. Tu má přiřazenu 6to4 směrovač, který musí být připojen jak k IPv4 Internetu, tak ke koncové IPv6 síti a procházejí jím veškerá data přepravovaná 6to4. Typické uspořádání vypadá tak, že 6to4 realizuje přístupový směrovač sítě a dotyčnou IPv4 adresou je adresa tohoto směrovače. Roli 6to4 směrovače může hrát celkem libovolný stroj v síti, přístupový směrovač bývá pro tuto úlohu ideálním kandidátem. [1]

6to4 na základě dané IPv4 adresy vytvoří IPv6 prefix délky 48 bitů pro celou síť. Začíná hodnotou 2002::/16 a dalších 32 bitů tvoří IPv4 adresa přístupového směrovače. Vznikne tak prefix standardní délky, který umožní adresovat počítače v síti obvyklým způsobem. Strukturu adres používajících 6to4 znázorňuje Obrázek: 5. Kdyby například 6to4 směrovač měl IPv4 adresu 147.230.7.23, vytvořil by pro svou IPv6 síť prefix 2002:93e6:717::/48.

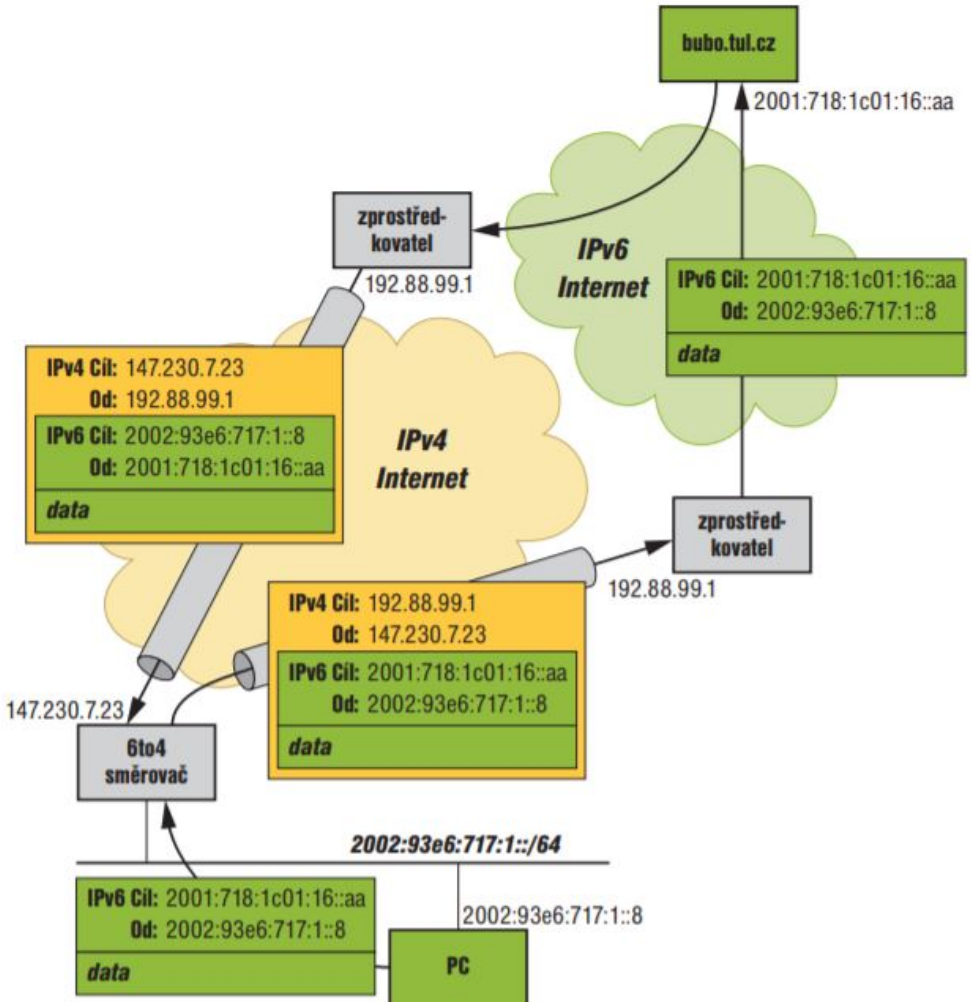


Obrázek 5: Struktura 6to4 adresy, přejato z:[1, str.258].

Dotyčná IPv6 síť používá 6to4 prefixy jako jedny ze svých IPv6 prefixů. Směrovač implementující 6to4 pak do vnitřní sítě ohlašuje směrovací informaci, že přes něj vede cesta k síti 2002::/16. Datagramy adresované do jiné 6to4 sítě budou proto předány k doručení jemu. Když se tak stane, vezme si z cílové IPv6 adresy IPv4 adresu 6to4 směrovače vzdálené sítě a datagram mu automaticky tuneluje. Jako zdrojovou adresu použije IPv4 adresu svého odchozího rozhraní. Tunel není sestaven trvale, IPv6 paket se podle tunelovacích pravidel jednoduše zabalí do IPv4 datagramu a odešle do Internetu, 6to4 směrovač si pro něj nemusí ukládat žádné stavové informace. Významnou předností 6to4 je jeho minimální náročnost. Není potřeba žádného zásahu do IPv4 směrování a jen minimální zásah do IPv6 směrování (ohlašování 2002::/16 do lokální sítě, zbytek se odehrává na bázi automatických tunelů). Nevyžaduje žádné konfigurované tunely, k životu mu stačí jen přístupový směrovač podporující 6to4. Největší problém vzniká při styku s přirozeným IPv6 světem – když spolu potřebují komunikovat stroje, z nichž jeden má pouze 6to4 adresu a druhý jen nativní IPv6 adresu. V takovém případě musí v Internetu existovat

alespoň jeden zprostředkovatel (relay router). Potřebuje alespoň jedno 6to4 rozhraní a alespoň jedno nativní IPv6 rozhraní s plnohodnotným připojením. Jeho pozice vůči nativnímu IPv6 světu je celkem jednoduchá – obvyklými směrovacími mechanismy (BGP) ohlašuje, že jeho prostřednictvím je dosažitelná síť s prefixem 2002::/16. Je-li takových směrovačů více, standardním způsobem se posoudí, který z nich je pro daný IPv6 uzel nejvýhodnější.[1]

Výměna dat mezi 6to4 a nativním IPv6 má bohužel silný sklon k asymetrii. Datagramy směřující do nativní IPv6 sítě bude předávat držitel výběrové adresy 192.88.99.1 nejbližší 6to4 odesilateli, zatímco datagramy v protisměru budou předány zprostředkovateli, který je nejblíže protějšímu stroji. [1] Situaci ilustruje Obrázek: 6.



Obrázek 6: Výměna dat mezi 6to4 a nativním IPv6, přejato z:[1, str.259].

4 Příprava pro praktickou ukázku

Praktická úloha porovnává implementační možnosti aplikačního firewallu na platformě Linux proti aplikačnímu firewallu na platformě Cisco ASA. Jako zadání aplikační ochrany byly plněny následující požadavky:

- Firewall má zabránit HTTP požadavkům nesoucí znaky SQL injekce proti webovým serverům v interní síti.
- Firewall má blokovat nahrávání souborů s příponou mediálních souborů (mp3.,.mp4,.avi) pomocí protokolu. FTP
- Firewall má blokovat uživatelům z vnější sítě přejmenovávat soubory na FTP serverech ve vnitřní síti.

Obecné schéma topologie sítě vykresluje: Obrázek 7, podrobné schéma včetně adresace lze nalézt v příloze C, Obrázek 41.

Hardwarové a Softwarové vybavení laboratoře

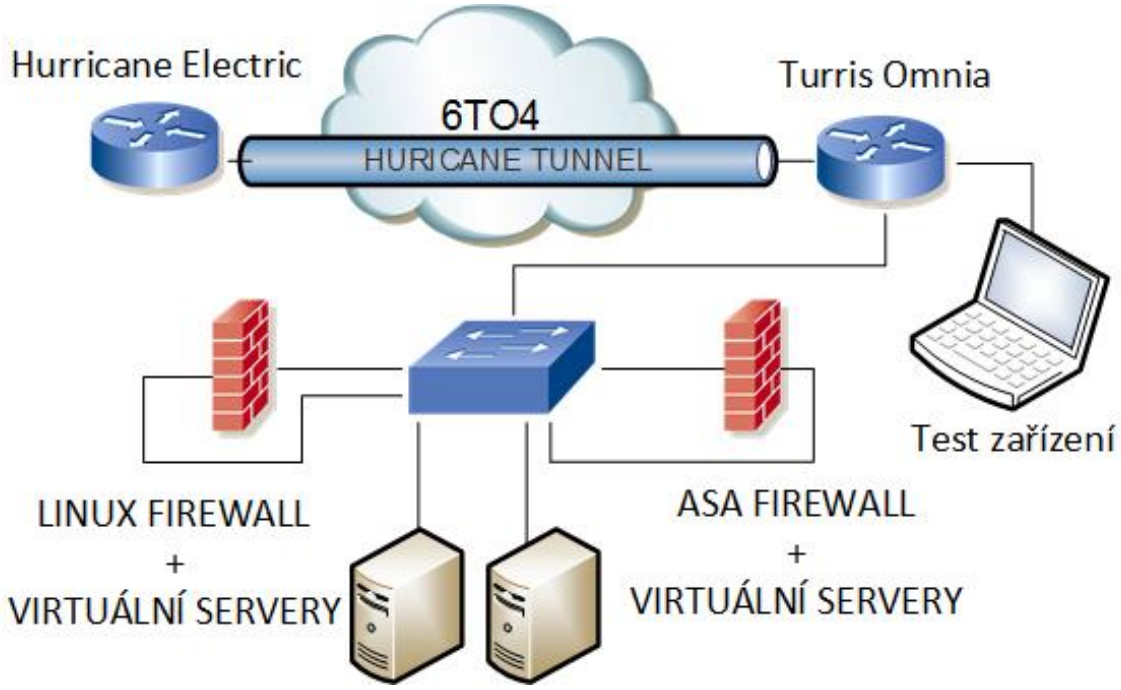
- **ASA 5506-X Firepower**
  8 x 1 Gigabit Ethernet (GE)
  Celkových 4 GB RAM
  PCU: Atom C2000 series 1250 MHz, 1 CPU (4 jádra)
  ASA: 1843 MB RAM, 1 jádro CPU
  FirePower: 2253 MB RAM, 1 CPU (3 jádra)
  50 GB mSata SSD DISK
  Crypto Accelerator: Cavium Octeon III CN7020 2 core 1.2GHz

- **Linux Firewall**
  Lenovo ThinkCentre A58
  CPU: Intel Core2 Duo E7500 2936 MHz
  80GB HDD
  Realtek RTL8111 PCIE Gigabit Ethernet Controller
  Marvell 88E8057 PCIE Gigabit Ethernet Contoller
  Intel 4 Series Integrated Graphics
  Ubuntu 18.04.1 TLS

- **Přepínač**
  Cisco Catalyst 2950
  24x Ethernet 100Base-TX

- **Směrovač**
  Turris omnia 1G
  1 GB DDR3 RAM
  5x Gbit port
  1x Gbit port SFP
  1.6 GHz ARMv7 CPU (2 jádra)

  Host pro virtuální servery
  Lenovo IdeaPad Z560
  Intel Core i3 CPU M370 2,4 GHz
  8 GB RAM

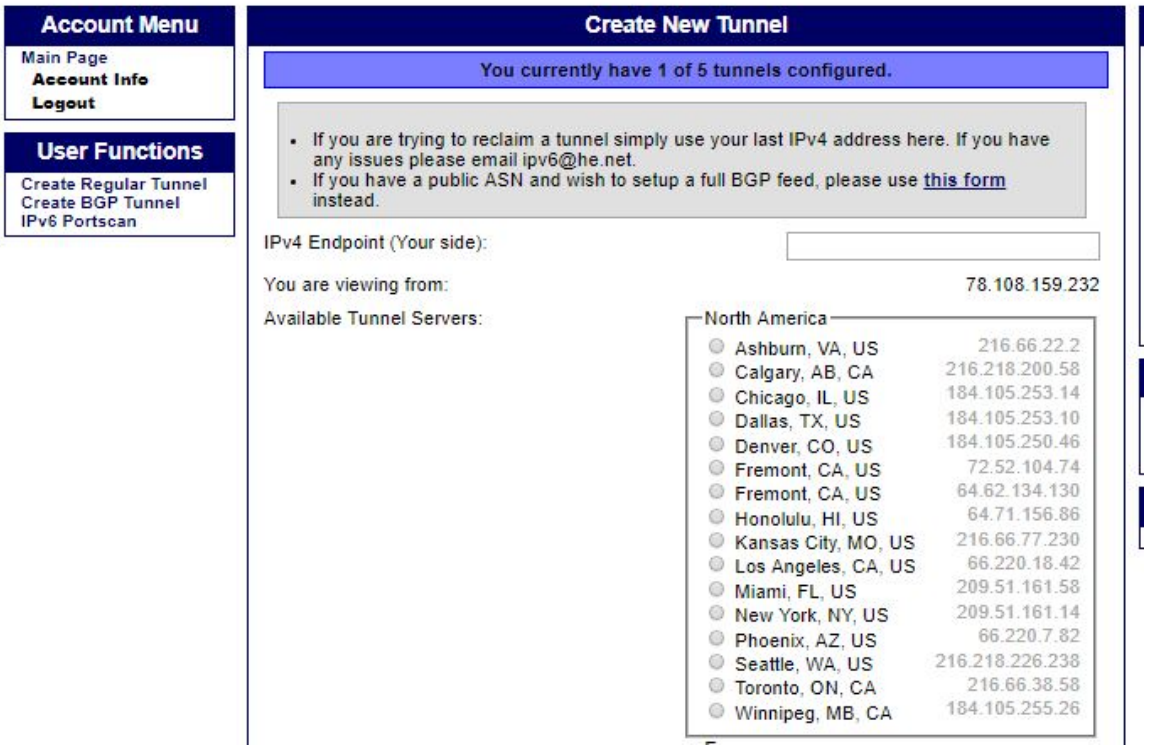


Obrázek 7: Obecné schéma laboratorní úlohy.

Pro simulaci a testování SQL injekce byl použit předinstalovaný obraz projektu Seed Labs. Projekt pod vedením profesora Wenliang Du, Syracuse University vznikl v roce 2002, a poskytuje studijní materiály a více než 30. laboratorních cvičení v oboru počítačové bezpečnosti. Obraz systému je pro studijní účely dostupný z: [24]. FTP servery byly implementovány na operačním systému Lubuntu Alternate Version 18.10. Veškeré servery byly provozovány virtuálně pomocí sofware Oracle Virtual box, fyzicky na osobním notebooku Lenovo IdeaPad Z560.

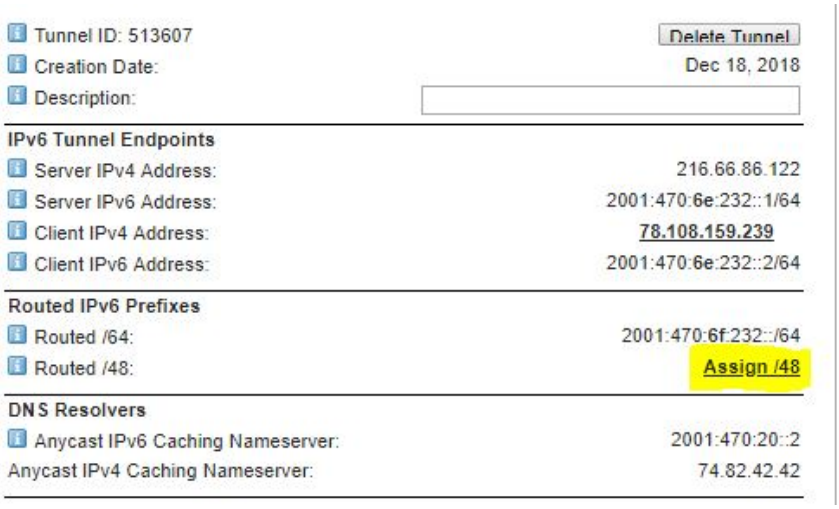
4.1 IPv6 konektivita a konfigurace 6to4 na OpenWRT

Jelikož IPv6 stále nepatří mezi základní služby poskytovatelů internetu, pro pořízení IPv6 internet konektivity byl využit 6to4 tunel společnosti Huricane Electric. Pro vytvoření tunelu je potřeba mít alespoň jednu veřejnou IPv4 adresu, pokud je adresa za NATem, musí být povolen přenos protokolu IP41. Po registraci na webu `https://www.tunnelbroker.net/` se nachází v levém horním rohu možnosti vytvoření obyčejného tunelu nebo tunelu s BGP. Společnost poskytuje možnost vytvoření tunelu s plným BGP peeringem na úrovni internetu (je třeba registrace autonomního systému). Pro laboratorní úlohu postačí obyčejný tunel. Další dvě věci, které je třeba při vytváření tunelu, znát je IPv4 adresa koncového bodu. Následně je potřeba vybrat tunel server, který je nejblíže viz.: Obrázek 8.



Obrázek 8: Formulář pro vytvoření tunelu.

Potvrzením formuláře proběhne přesměrování zpět na domovskou stránku, kde jsou vidět parametry připraveného tunelu. Huricane Electric automaticky poskytuje /64 směrovaný prefix na každý tunel. Pokud je pro tyto účely potřeba více adres. Kliknutím na jméno tunelu si je možno jedním tlačítkem „Assign /48“ přiřadit plně směrovaný /48 prefix pro tuto laboratorní úlohu viz.: Obrázek 9.



Obrázek 9: Žádost o /48 směrovaný prefix.

OpenWRT konfigurace

Pro spuštění tunelu na OpenWRT směrovači jsou třeba dva kroky:

1. Vytvoření 6to4 rohraní:

Rozhraní se vytváří v souboru /etc/config/network

```
config interface 'wan6in4'
        option proto '6in4'
        option mtu '1480'
        option peeraddr '216.66.80.30'
        option ip6prefix '2001:470:736e::/48'
        option ip6addr '2001:470:1f0a:a49::2/64'
```

Výpis 1: Konfigurace rozhraní pro 6to4

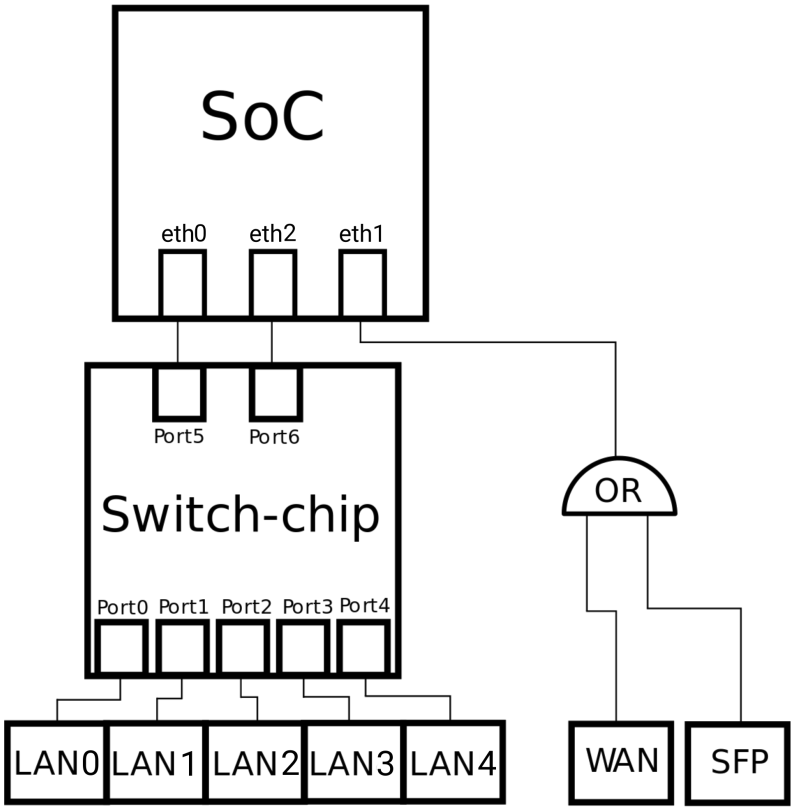
Firewall pravidlo definujeme /etc/config/firewall

```
config rule
        option name 'wan6in4'
        option family 'ipv4'
        option src 'wan'
        option src_ip '216.66.80.30'
        option proto '41'
        option target 'ACCEPT'
```

Výpis 2: Konfigurace firewallu pro povolení IP41

Tato konfigurace se připojí k HE a je možné v grafickém rozhraní vidět dané rozhraní online, také je možnost otestovat dostupnost IPv6 adres v internetu. Pro kompletní funkci laboratoře je třeba další konfigurace.

Dle schématu interního zapojení síťových sběrnic uvnitř směrovače Turris Omnia bylo třeba vytvořit novou VLAN přes kterou bylo vytvořeno směrování proti testovaným firewallům.



Obrázek 10: Interní zapojení sběrnic směrovače Turris Omnia, převzato z:[8].

Konfigurace leží v souboru /etc/config/interfaces

```
config switch
        option name 'switch0'
        option reset '1'
        option enable_vlan '1'

config switch_vlan
        option device 'switch0'
        option vlan '1'
        option vid '1' // vlan 1 používaná pro interní síť
```

```
        option ports '1 2 3 5t' // 1,2,3 jsou neznačkované port 5 je
    značkovaný

config switch_vlan
        option device 'switch0'
        option vlan '3'
        option vid '3'
        option ports '0 5t' // vlan 3 je připojena pouze na port 0
    neznačkovaně a port 5 značkovaně
```

Další částí konfigurace je vytvoření rozhraní L3 pro danou podsíť. /etc/config/interfaces

```
config interface 'DMZv6'   // název rohraní je DMZv6
        option proto 'static' // statická adresa
        option ifname 'eth0.3' //podrozhraní eth0 s vlan 3
        option ip6addr '2001:470:736e:a::1/64' // ipv6 adresa rozhran
    í
        option ip6prefix '2001:470:736e:a::/64' // prefix dané podsít
    ě
```

Výpis 3: Konfigurace L3 rozhraní

Poslední část konfigurace v /etc/config/interfaces obsahuje vytvoření statického směrování podsítí proti firewallům.

```
config rule
      config route6
        option interface 'DMZv6'
        option target '2001:470:736e:b::/64' //interní prefix ASA
    firewallu
        option gateway '2001:470:736e:a::2' //adresa venkovního
    rozhraní ASA

config route6
        option interface 'DMZv6'
        option target '2001:470:736e:d::/64' //interní prefix Linux
    firewallu
        option gateway '2001:470:736e:a::3' // adresa venkovního
    rozhraní Linux
```

Výpis 4: Konfigurace statického směrování

Dále je třeba definovat firewall a pravidla pro dané DMZv6 rozhraní. Konfigurace leží v souboru: /etc/config/firewall

```
config zone    // přidání rohraní do wan zóny
        option name 'wan'
        option mtu_fix '1'
        option input 'REJECT'
        option network 'wan wan6 wan6in4'
        option output 'ACCEPT'
        option masq '1'
        option forward 'REJECT'

config rule //povolení veškerého IPv6 provozu z wan na podsíť Ubuntu
        option target 'ACCEPT'
        option src 'wan'
        option dest 'DMZv6'
        option family 'ipv6'
        option proto 'all'
        option dest_ip '2001:470:736e:d::/64'
        option name 'UbuntuFW'

config rule  //povolení veškerého IPv6 provozu z wan na podsíť ASA
        option target 'ACCEPT'
        option src 'wan'
        option name 'DMZ'
        option family 'ipv6'
        option proto 'all'
        option dest 'DMZv6'
        option dest_ip '2001:470:736e:b::/64'

config rule //povolení veškerého provozu z LAN do sítí v zóně DMZv6
        option enabled '1'
        option target 'ACCEPT'
        option src 'lan'
        option dest 'DMZv6'
        option name 'Testing'
        option proto 'all'
```

Výpis 5: Konfigurace L3 rozhraní

4.2 Příprava serverů pro SQL injekci

Pro souběžný běh více systémů na jednom fyzickém hardwaru byl využit moderní open-source virtualizační nástroj Oracle Virtual Box. Software je dostupný zdarma viz.:[23]. Instalace probíhá podle standardního windows instalačního procesu. Předinstalovaný obraz systému SEEDUbuntu16.04, lze najít na webu SEED LABS projektu viz.:[24]. Po stažení a rozbalení souboru s obrazem systému otevřeme obraz disku následujícím postupem: V programu Virtual box se kline na modrou ikonu „nový“ v následně otevřeném okně se vyplní pole názvu počítače dle vlastního uvážení, typ systému zvolíme Linux, verze Ubuntu (64bit). Tlačítko „další“ nás posune k nastavení velikosti paměti RAM pro daný systém. Zde je třeba ponechat výchozí hodnotu 1024 MB a posunout se dále. Následující nabídka obsahuje výběr pevného disku pro Virtuální stroj (VM - z ang. Virtual machine). Volí se použití existujícího souboru s virtuálním pevným diskem. Klikem na ikonu složky je navigováno stromovou strukturou k adresáři se staženým a rozbaleným virtuálním diskem ve formátu .vdmk. Tlačítko vytvořit dokončí vložení VM do Virtual boxu. Systém je připraven k použití a dalším úpravám.

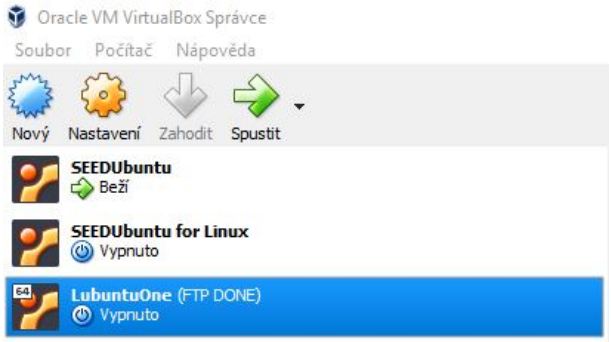
Jelikož výchozí chování síťového rozhraní pro jednotlivé VM je NAT, přičemž je každý systém umístěn do vlastní oddělené privátní sítě, jehož komunikace se při odchozích spojeních maskuje za adresu rozhraní hostujícího operačního systému. VM je proto z hlediska sítě neodhalitelná a navázat spojení z venčí je při tomto nastavení principiálně nemožné. Pro umožnění obousměrné komunikace z vnější sítě je nutné nastavit síťové rozhraní VM jako síťový most k síťové kartě, která je poté fyzicky připojená do síťové infrastruktury a logicky propojena s vnitřním rozhraním firewallu viz.: Obrázek: 7. Pro možnost tohoto nastavení je třeba kliknout v seznamu virtuálních systémů na příslušnou VM. Klikem na ikonu ozubeného kolečka se otevře okno nastavení. V menu se přejde pod záložku síť a tím se aktivuje síťová karta. Z možností připojení se vybere připojit k síťovému mostu, v možnostech „název“ je vybrána vhodná síťová karta (interní/usb), která je připojena kabelem k přepínači pod vhodnou VLAN. Po potvrzení je VM připravená ke spuštění.

Virtuální server se spustí označením příslušné VM a následným klikem na tlačítko spustit. Ikona stavu pod názvem VM se změní ze stavu vypnuto na stav běží viz.: Obrázek 11. Při označení spuštěné VM se ikona spustit změní na ikonu zobrazit. Touto ikonou se přepne na pracovní plochu příslušného operačního systému.

Systém SeedUbuntu je předinstalován a přednastaven pro použití mnoha simulací současně. Je doporučeno nespouštět aktualizaci, jelikož by aktualizace mohla opravit bezpečnostní chyby využívány v laboratorních úlohách.

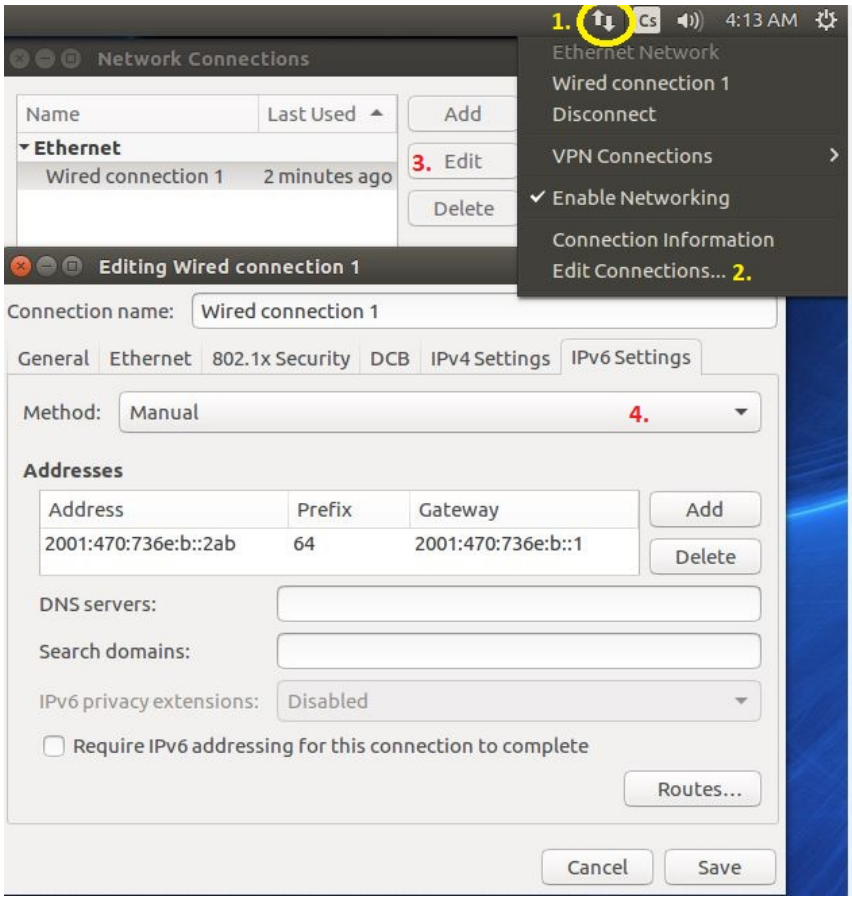
Pro demonstrování SQL injekce v IPv6 je ale třeba udělat několik úprav pro správnou funkci v právě používaném prostředí.

- **Nastavení IPv6 rozhraní** Pro nastavení adresy je systémem poskytnuto příjemné uživatelské rozhraní. Postup nastavení vyobrazuje Obrázek: 12. Přes ikonku šipek na horní navigační liště se nachází funkce editace spojení. Vyberáním editovat spojení ethernet, a



Obrázek 11: Stav běžícího virtuálního serveru.

vybráním záložky IPv6 nastavení. Zvolením manuální metody volby adresy a do políček s adresou se vyplní zvolená adresa přidělena z dané IPv6 podsítě.



Obrázek 12: Nastavení IPv6 adresy.

- **Nastavení webového serveru** Přednastavená konfigurace apache2 využívá funkci VirtualHost, která umožňuje běh několika webů na jednom serveru a zároveň na jednom portu současně. Princip spočívá v definici spojení URL s adresářem obsahujícím pří-

slušné konfigurační soubory dané webové stránky. Tato konfigurace probíhá editací souboru /etc/apache2/sites-available/000-default.conf. Je třeba nahradit ServerName v sekci s SQLInjection webem, za náš vlastní hostname, který byl vytvořen pro přístupnost webu z veřejného internetu. DNS záznam byl vytvořen pod již existující doménou, které jsem vlastníkem.

```
<VirtualHost *:80>
        ServerName http://www.inject-asa.stibicweb.cz
        DocumentRoot /var/www/SQLInjection
</VirtualHost>
```

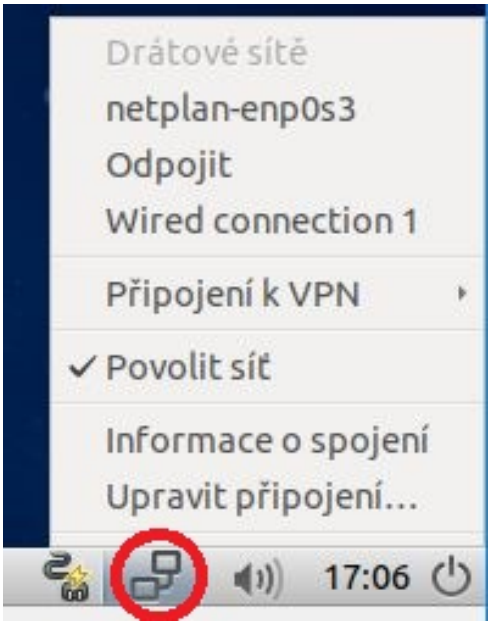
Výpis 6: Konfigurace apache2

Celý postup je potřeba zopakovat ještě jednou, pouze s výměnou adresního rozsahu a hostname určený pro použití v síti s Linux firewallem. V případě potíží je celá uživatelská dokumentace k obrazu serveru je dostupná z odkazu viz.:[25].

4.3 Příprava FTP serverů

Pro menší paměťovou náročnost a větší volnost při instalaci balíčků a aktualizací byl pro simulaci úlohy s FTP zvolen systém Lubuntu Alternate, dostupný z webu:[26]. Po stažení obrazu systému je postup podobný vkládání již předinstalovaného systému. Instalace započne kliknutím na ikonku „Nový“ v levém rohu programu Virtual Box. Pro název počítače je vybrán libovolný název, typ systému je zvolen Linux, verze: Ubuntu(64-bit). V druhém kroku je možnost paměť snížit na 512 MB. Dále při výběru pevného disku, je zvolena možnost vytvoření nového virtuálního disku. Typ souboru s pevným diskem je možnost ponechat ve výchozí hodnotě VDI. V posledním kroku vytváření disku probíhá volba velikosti disku pro instalovanou VM. Pro dostatečný prostor pro ukládání souborů je zvýšen prostor na 15 GB. Před celkovým spuštěním se přepne síťový adaptér do režimu most, stejně jak bylo provedeno při instalaci VM Seed Labs. Při následném prvním spuštění je uživatel vyzván k volbě bootovacího disku. Přes ikonku složky opět je navigováno stromovou strukturou do adresáře se staženým obrazem Lubuntu.

Pokračuje standardní proces instalace systému, až do stavu systému s připravenou plochou k použití. Pokud je plocha připravena k použití, je třeba znovu nastavit IPv6 rozhraní. Postup v Lubuntu je téměř shodný s nastavením prováděným v systému Ubuntu pro Seed Labs, rozdílná je pouze ikona reprezentující nastavení sítě, která se nachází na Obrázku:13. Detail cesty k nastavení je shodný s tím, jež se nachází na Obrázku:12. Oproti nastavení prováděném pro Seed Lab, pro FTP servery je potřeba nastavit i veřejnou DNS. Je možno použít IPv6 alternativu k nejznámější DNS od google 8.8.8.8 > 2001:4860:4860::8888.



Obrázek 13: Nastavení sítě v Lubuntu.

Po dokončení konfigurace síťového rozhraní se začne s instalací FTP serveru. Stiskem klávesové kombinace ctrl+alt+t se otevře Linux terminál. V terminálu se spustí instalace s danou posloupností příkazů:

```
sudo apt-get update // aktualizace instalačnních balíčků
sudo apt-get upgrade // instalace aktualizací zastaralých verzí
  software
sudo apt-get install vsftpd  // instalace open-souce ftp serveru
```

Výpis 7: Instalace FTP serveru

Příkazem sudo nano /etc/vsftpd.conf je editován konfigurační soubor. Provede se modifikace příslušných parametrů.

```
anonymous_enable=NO // Zakáz neidentifikovaných uživatelů
local_enable=YES // povolení ftp pro uživatele s účtem v systému
write_enable=YES // povolení zápisu
chroot_local_user=YES //omezení přístupu na domovský adresář
listen=NO // vypnutí IPv4
listen_ipv6=YES // zapnutí IPv6
```

Výpis 8: Nastavení ftp serveru v /etc/vsftpd.conf

Editaci je ukončena klávesovou zkratkou ctrl+x, poté potvrzeno uložení změn klávesou „y“, uloženo stiskem klávesy „enter“. Posledním krokem je restart aplikace příkazem:

```
sudo service vsftpd restart
```

Následně je možno ověřit naslouchání serveru na IPv6.

```
sudo netstat -tunlp | grep :21
                ///// výsledek //////
tcp6    0     0 :::21       :::*    NASLOUCHÁ 5881/vsftpd
```

Výpis 9: Ověření správné funkce serveru

FTP server je nyní nakonfigurován k použití systémových účtů k ověření přístupu. Nový uživatel je v systému vytvořen pomocí příkazu:

```
sudo adduser tom // vytvoření uživatele
```

Současně je tento server použit i pro test propustnosti HTTP provozu. Instalace webového serveru probíhá pomocí příkazu:

```
sudo apt-get install apache2
```

Pro testování propustnosti je potřeba jeden soubor o dostatečné velikosti, vzhledem k času potřebnému k shromáždění všech potřebných údajů o prováděném přenosu. Při maximální propustnosti sítě 100 Mbit/s je velikost 1 GB dostatečné množství dat. K testování byl použit obraz systému modulu firepower, který svou velikostí odpovídá tomuto požadavku. Soubor je třeba uložit do adresáře /var/www/html/. Poté změnit práva souboru a povolit čtení všem, jinak bude soubor dostupný pouze z lokálního systému. Změnu potřebných práv provedeme příkazem:

```
sudo chmod 755 test.img
```

Jelikož budou potřebova dva oddělené servery s téměř totožným nastavením, druhý server není nutné instalovat touto časově náročnou formou. Může být využita funkce Virtual Boxu klonovat existující server a změnit pouze IP adresu a nastavit most k jiné síťové kartě. Před samotným klonováním musí být systém klonované VM vypnut. Možnost klonování se nachází při označení příslušné VM pravým tlačítkem myši. Po stisknutí tlačítka klonovat se vyplní nový název pro VM a zaškrtne se požadavek pro reinicializaci MAC adresy všech síťových karet. Dále se je potřeba se proklikat až k vytvoření nového klonu s ponecháním všech možností ve výchozím stavu. Nastavení sítě klonované VM změní se dle postupu z předešlého textu.

5 Adaptive Security Appliance (ASA)

Adaptivní zabezpečovací zařízení řady Cisco ASA 5500 poskytují robustní sadu vysoce integrovaných bezpečnostních služeb pro malé a střední podniky. Řada Cisco ASA 5500 není pouze čistým hardwarovým firewallem. Stručně řečeno, Cisco ASA je bezpečnostní zařízení, které kombinuje funkce brány firewall, antivirové ochrany, prevence narušení (IPS) a funkce virtuálních privátních sítí (VPN). Poskytuje aktivní ochranu před hrozbami, která zastaví útoky dříve, než se rozšíří po síti. Cisco ASA firewall je vlastně celý balíček. S přídavnými moduly, které nabízejí celou řadu funkcí, je možné získat ještě více funkcí zabezpečení [13].

S vylepšením v řadě 5500-X Cisco nabízí bránu firewall nové generace. Jedná se o firewall Cisco ASA se službami FirePOWER, které jsou k dispozici na zařízeních Cisco ASA 5500-X Series a ASA 5585-X Adaptive Security Appliances. Díky tomuto řešení lze získat osvědčenou ochranu Cisco ASA firewall v kombinaci s špičkovou orchanou Sourcefire a pokročilou ochranou proti malwaru v jediném zařízení [16].

Obrázek 14: Cisco ASA 5506-X

Firewall v základní verzi podporuje přesně definovanou sadu inspekcí protokolů (CTIQBE, DCERPC, DNS, DNS, FTP, H.323, HTTP, ICMP, IM, IPsec Pass Through, MGCP, MMP, NetBIOS, PPTP, RADIUS, RSH, RTSP, SCTP, SIP, SMTP, SNMP, SQL*Net, STUN, TFTP, XDMCP, VXLAN viz.:[14]). Pro tyto protokoly jsou definovány možnosti inspekce a blokace na základě vymezených parametrů a obsahu v aplikačních datech. Pro potřebu blokace nebezpečných signatur nad aplikačními daty i mimo tyto protokoly je pro firewally ASA řady 5000 možnost instalace hardwarového modulu IPS. Databáze signatur pro Cisco ASA IPS je dostupná online, viz.:[15]. Databáze je poté možné nechat aktualizovat automaticky nebo provádět aktualizaci signatur ručně. Prodej hadwarové IPS byl ukončen dne 26. dubna 2018 [15], databáze je ale stále aktualizovaná o nové signatury. Produkt ASA IPS byl nahrazen funkcí Firepower v řadě ASA 5500-X. Firepower IPS provádí kontrolu více než 4000 koncových aplikací. Zároveň je IPS spojena s projektem SourceFire, a tudíž přidává signatury z největší open-source IPS databáze Snort [4]. Dále umožňuje kontrolu a klasifikaci stovek milionů URL do skupin podle obsahu a mnoho dalšího viz.:[16]. Z licenčních důvodů se diplomová práce nebude filtrací Firepower dále zabývat.

5.1 Příprava ASA firewallu

Po spuštění zařízení ve výchozím stavu je prvotní konfigurace možná pouze pomocí sériové konzole. Obvyklým prvním krokem je tedy tvorba lokálního účtu a povolení vzdálené správy pomocí SSH / ASDM (Adaptive Security Device Manager). Zařízení má pro tyto účely dedikovaný hardwarový management port. V praxi se management firewallů povoluje právě pouze z těchto dedikovaných rozhraní. Využití tohoto portu má výhody nejen co se týče bezpečnosti, protože odděluje management síť od uživatelské, ale také v případě přetížení zařízení, a při selhání vnitřních procesů mají tyto hardwarové management porty vlastní microprocesor, který umožňuje přístup k diagnostice i v případě, kdyby se SSH proces na obyčejném rozhraní vůbec nespustil. V laboratorním prostředí je potřeba nastavení provádět z vnitřní nebo školní sítě, proto toto dedikované rozhraní nebude použito.

Úvodní konfigurace tedy bude vypadat takto:

```
ASAv6(config)# username uzivatel password HesloVSB privilege 15
// uživatel s nejvyššími právy

ASAv6(config)# interface GigabitEthernet1/4 //vytvoření rozhraní pro
    vnitřní síť
ASAv6(config-if)# nameif inside
ASAv6(config-if)# security-level 100
ASAv6(config-if)# ipv6 address 2001:470:736e:b::1/64
ASAv6(config-if)# ipv6 enable
ASAv6(config-if)# ipv6 nd managed-config-flag


ASAv6(config)#asdm image disk0:/asdm-741.bin //cesta k souboru asdm

ASAv6(config)#user-identity default-domain LOCAL // Ověření identit v
     lokální db
ASAv6(config)#aaa authentication http console LOCAL // Ověření účtů
    pro ASDM z lokální db
ASAv6(config)#aaa authentication ssh console LOCAL //Ověření účtů pro
     SSH z lokální db
ASAv6(config)#http server enable  // Zapnutí ASDM serveru
ASAv6(config)#http 2001:470:736e:b::/64 inside //ASDM pro interní síť
ASAv6(config)#http 2001:718:1001::/48 outside //ASDM pro prefix VŠB
ASAv6(config)#ssh 2001:470:736e:b::/64 inside //SSH pro interní síť
ASAv6(config)#ssh 2001:718:1001::/48 outside //SSH pro prefix VŠB
```

```
ASAv6(config)# interface GigabitEthernet1/8 // Vytvoření rozhraní pro
    internet
ASAv6(config-if)# nameif outside
ASAv6(config-if)# security-level 0
ASAv6(config-if)# no ip address
ASAv6(config-if)# ipv6 address 2001:470:736e:a::2/64
ASAv6(config-if)# ipv6 enable

ASAv6(config)# ipv6 route outside ::/0 2001:470:736e:a::1 //Výchozí
   route do internetu

ASAv6(config)#object-group service InGlobalServices
 // skupina služeb pro příchozí spojení
ASAv6(config-service-object-group)# service-object icmp6
ASAv6(config-service-object-group)# service-object tcp-udp
   destination eq www
ASAv6(config-service-object-group)# service-object tcp destination eq
    ftp

ASAv6(config)#object-group service OutGlobalServices
//skupina služeb pro odchozí spojení
ASAv6(config-service-object-group)# service-object ip
ASAv6(config-service-object-group)# service-object icmp6


ASAv6(config)#access-list outside_access_in extended permit object-
   group InGlobalServices any 2001:470:736e:b::/64
// povolení příchozího provozu

ASAv6(config)#access-list inside_access_in extended permit object-
   group OutGlobalServices 2001:470:736e:b::/64 any6
// povolení odchozího provozu
```

V kterémkoli prohlížeči tento nyní firewall sám nabídne stažení ASDM přes HTTPS na vnitřní adrese zařízení (V tomto případě:`https://[2001:470:736e:b::1]`).

ASDM je ale java aplikace a bývá obtížnější najít kompatibilní verzi javy, aby se aplikace spustila. V případě potíží na systému Windows je možno využít pro konfiguraci Linuxovou VM ve VirtualBoxu.

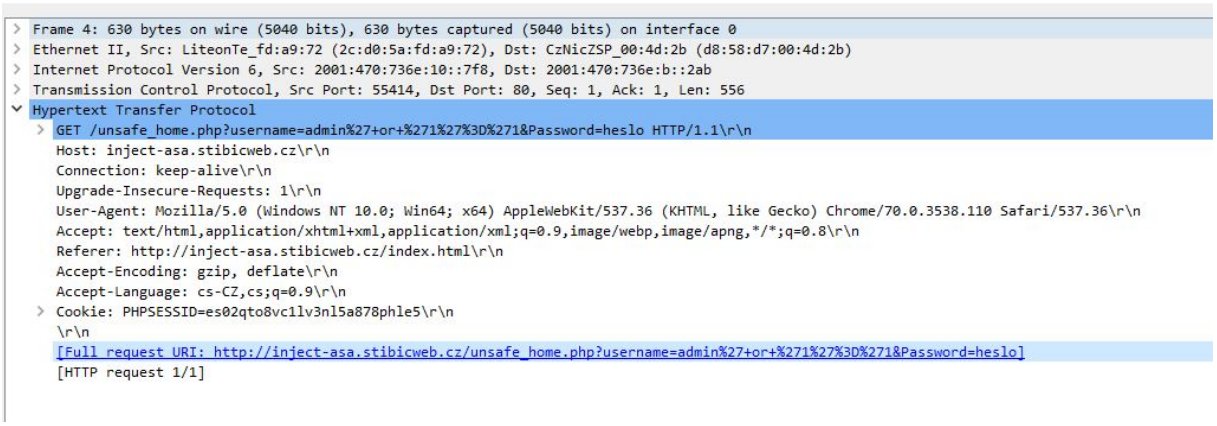
Postup instalace a spuštění ASDM na Linuxu je následující:

```
sudo apt-get install icedtea-plugin //* Instalace pro Java web start plugin.
javaws https://[2001:470:736e:b::1]/admin/public/asdm.jnpl //Stažení ASDM
```

Javaws vytvoří na ploše ASDM spouštěč, který má předkonfigurovanou a neměnnou IP adresu firewallu. Pokud budeme chtít ASDM připojit k jinému firewallu, musíme znovu použít javaws příkaz s modifikovanou adresou.

5.2 ASA aplikační firewall proti SQL injekci

Prvním krokem aplikačního filtru je znalost formátu aplikačních dat, které je potřeba blokovat. Popsat útok SQL injekce obecně je velice obtížné, existuje totiž nespočet možností jak přimět aplikaci aby pracovala tak, jak programátor nezamýšlel a nikdy by toto chování nedopustil. Pro ukázku obrany aplikačním firewallem byl vybrán útok, který s úspěchem obchází autentifikaci uživatele webu Seed Labs, ale také mnoha nejmenovaných webů veřejného internetu. K provedení tohoto útoku stačí v uživatelském jméně přihlašovacího formuláře webu zadat místo reálného jména výraz, který je SQL procesem v pozadí uznán jako vždy platný, a tudíž potvrzuje autentifikaci a povoluje přihlášení pod zvoleným uživatelem. Pro ukázku byl se známých SQL injekčních výrazů, testován výraz: „ admin' or '1'='1 " . Výraz touto metodou přihlásí uživatele admin a útočník tak získává práva číst a modifikovat záznamy celé databáze. Demonstraci přihlášení s užitím tohoto řetězce vyobrazuje příloha:D. Zachycením a analýzou daných dat pomocí programu Wireshark zjistíme umístění a formát dané HTTP zprávy, jíž klient žádá o přístup na na server.

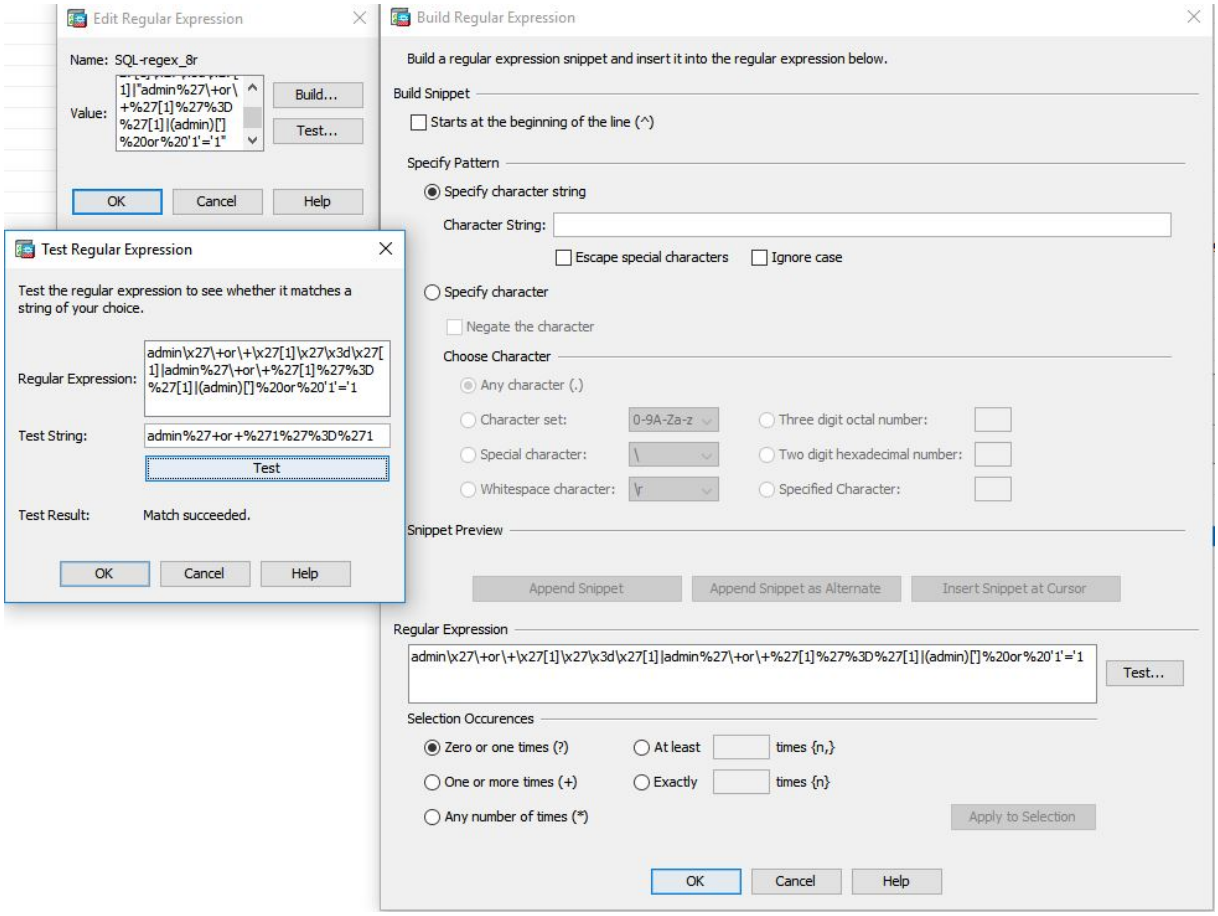


Obrázek 15: HTTP požadavek obsahující SQL injekci.

Dalším krokem aplikačního filtru jsou definice regulárních výrazů jejichž shodu bude firewall hledat pomocí inspekce aplikačních dat. Konfigurace regulárních výrazů se v ASDM nachází v

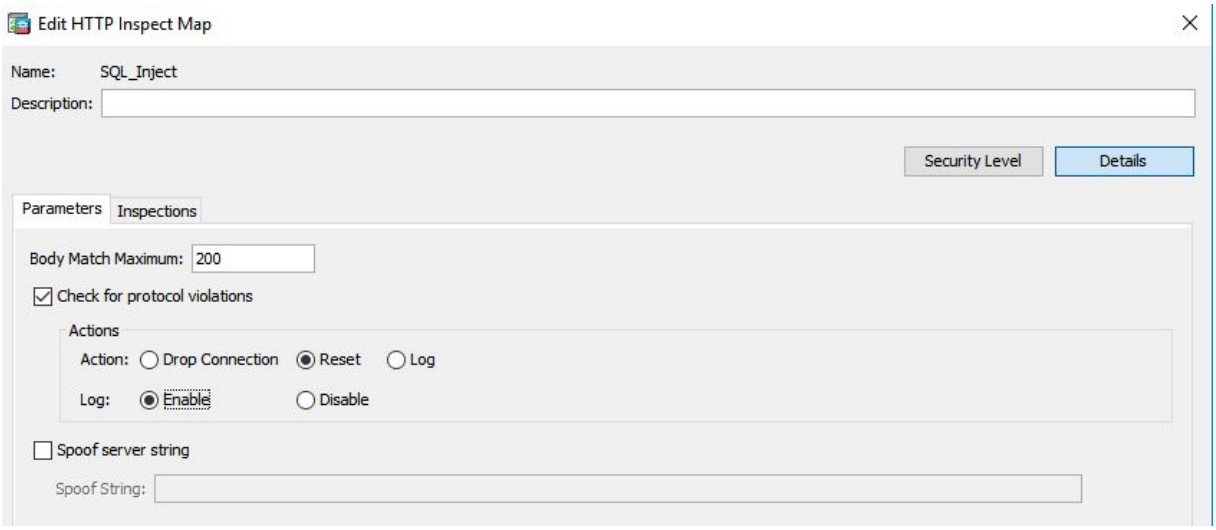
záložce Configuration> Firewall> Objects> Regular Expressions. Jak zobrazeno v Obrázku: 15, testovaný výraz se ukrývá ve zprávě HTTP get v položce Request URI. Našim úkolem je vytvořit takový regulární výraz, který by byl splněn pro výraz „ admin%27+or+%271%27%3D%271“ ve všech možných podobách a vyjádřeních. ASDM k tomu poskytuje nástroj i s možností otestovat potřebný řetězec, viz.:Obrazek 16. Nejdůležitějším poznatkem pro regulární výrazy týkající se URI jsou speciální znaky, např.:"%27"atd. Tyto znaky jsou hexadedimálním ASCII vyjádřením znaku „’“, které jsou přenášeny jako hexadecimální číslo, né jako trojice znaků „% + 2 + 7 “. Proto aby regulární výraz fungoval i v URI a nejen v textu je třeba definovat výraz i pro hexadecimální znaky pomocí syntaxe:"\x27"

Regulární výrazy se stejným účelem je možno shromáždit do jedné skupiny pod pojmem „Regular Expression Class-map“. Class-map pro regulární výrazy se vytváří ve stejném menu Configuration> Firewall> Objects> Regular Expressions, ve spodní části obrazovky. Pro účel detekce SQL injekce byla vytvořena class-map s názvem: SQL_INJECTION. Po vytvoření nás firewall vyzve k označení regulárních výrazů jenž má clas- map zahrnovat.



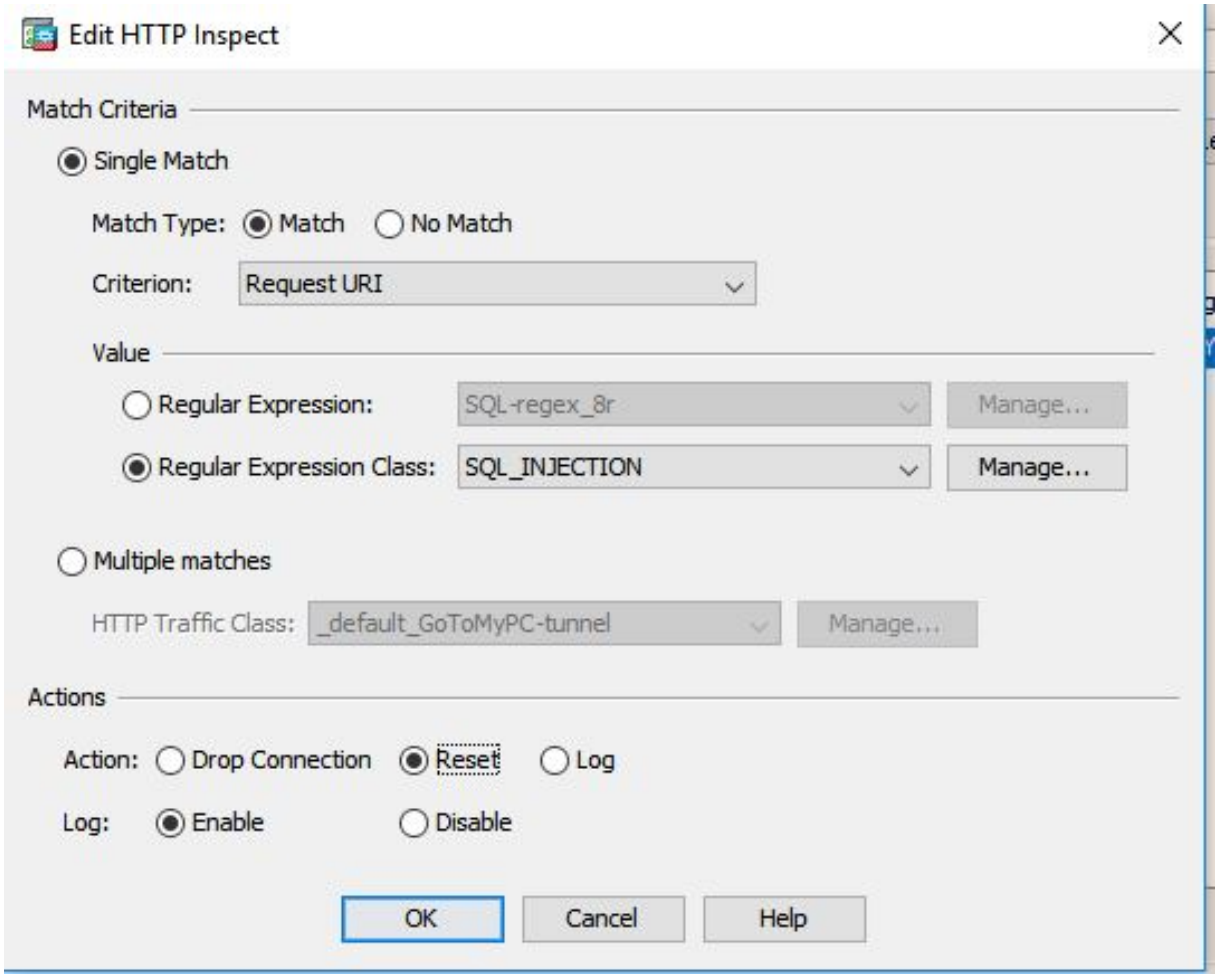
Obrázek 16: Nástroj pro tvorbu a testování regulárních výrazů.

Následujícím bodem konfigurace je definice inspekční mapy. Konfigurace probíhá v menu Configuration > Firewall > Inspect Maps. Firewall umožňuje definovat inspekci pouze pro několik základních protokolů(viz.:[14]). Přejitím pod záložku HTTP a kliknutím na tlačítko „ADD“ se vytvoří nová mapa. Tlačítkem detail se posune k bodu nastavení kontroly protokolu. Jinými slovy firewall kontroluje, zda-li je provoz opravdu HTTP a nepřenáší se pod daným protokolem, například FTP. Tímto nastává omezení pouze na kontrolu podporovaných protokolů. Tudíž není možné si vytvořit kompletně vlastní definici inspekce jiných protokolů. Ovšem pokud se provoz neshoduje s RFC protokolu firewall spojení ukončí pomocí TCP, RESET a vytvoří syslog zprávu o porušení RFC HTTP protokolu. Detail nastavení je vidět na Obrázku: 17



Obrázek 17: Detail nastavení kontroly konzistence protokolu.

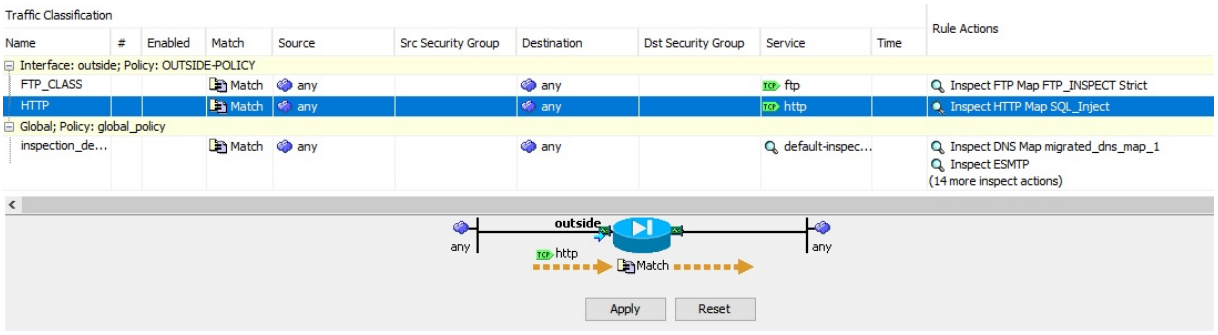
Záložka Inpections poté definuje, která část aplikačních dat se má kontrolovat, s jakým regulárním výrazem nebo class-mapou se má porovnávat, a jaká má být výsledná akce v případě nalezení definovaného výrazu. Z dřívější analýzy dat bylo zjištěno, že daný výraz se přenáší v HTTP URI. Zvolením „single match“ pravidla s kritériem shody Request URI a požadavkem shody definovaným jako Regular expression class: SQL_INJECTION viz.: Obrázek: 18. Pokud je nalezena shoda firewall ukončí spojení pomocí TCP, RESET a vytvoří syslog o dané události.



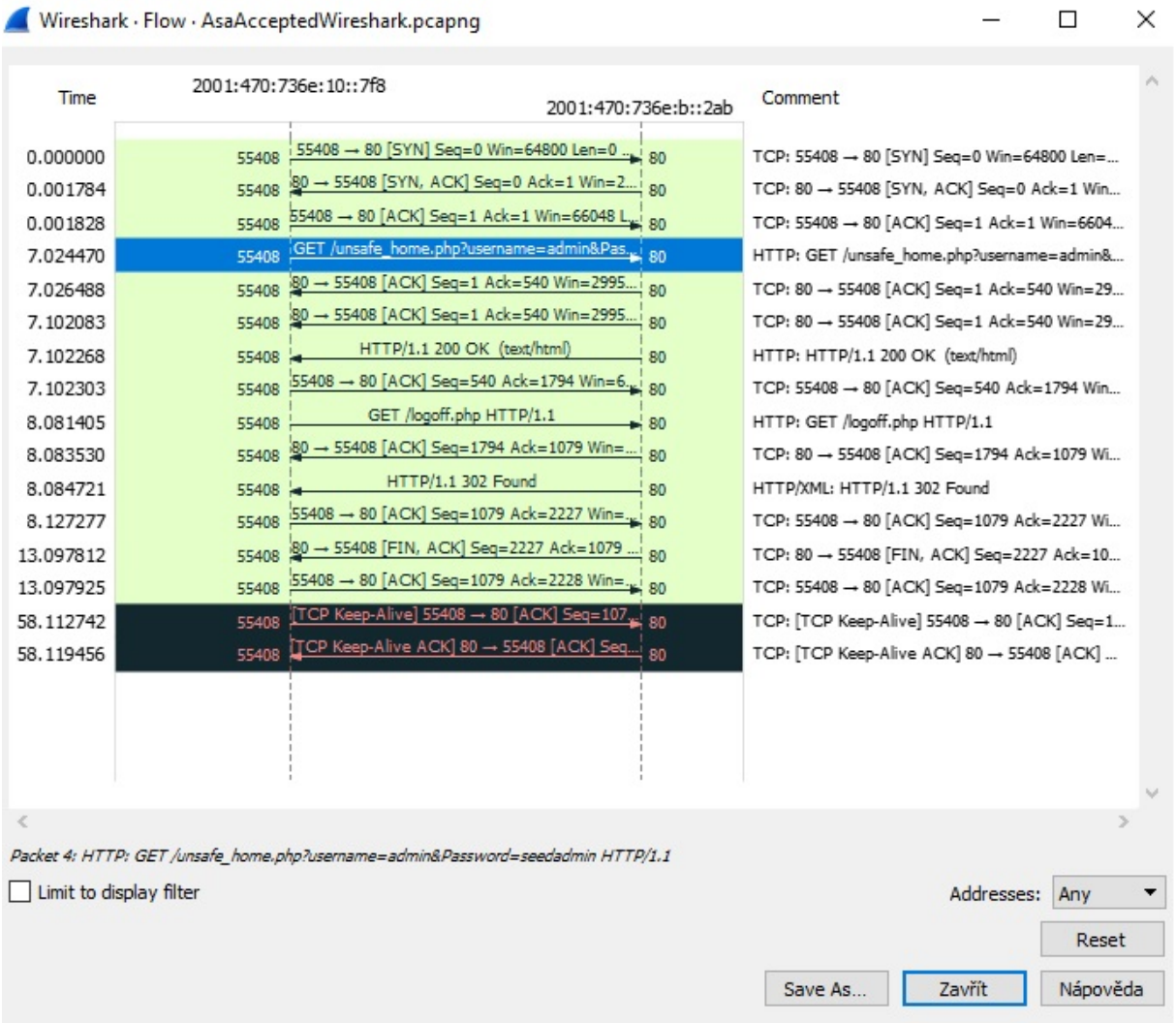
Obrázek 18: Nastavení inspekčního pravidla.

Posledním bodem konfigurace je použití vytvořené inspekční mapy. V menu Configuration > Firewall > Service Policy Rules se nastaví tzv. klasifikace provozu. Jedná se o definici pravidla, kdy je požadováno na použití dané mapy. Pro tuto definovanou úlohu byla zvolena aplikace inspekční mapy SQL_Inject na veškerý provoz procházející přes „OUTSIDE“ rozhraní splňující protokol HTTP na portu TCP/80. Při označení pravidla myší se zobrazí grafické vyjádření pravidla viz.: Obrázek: 19.

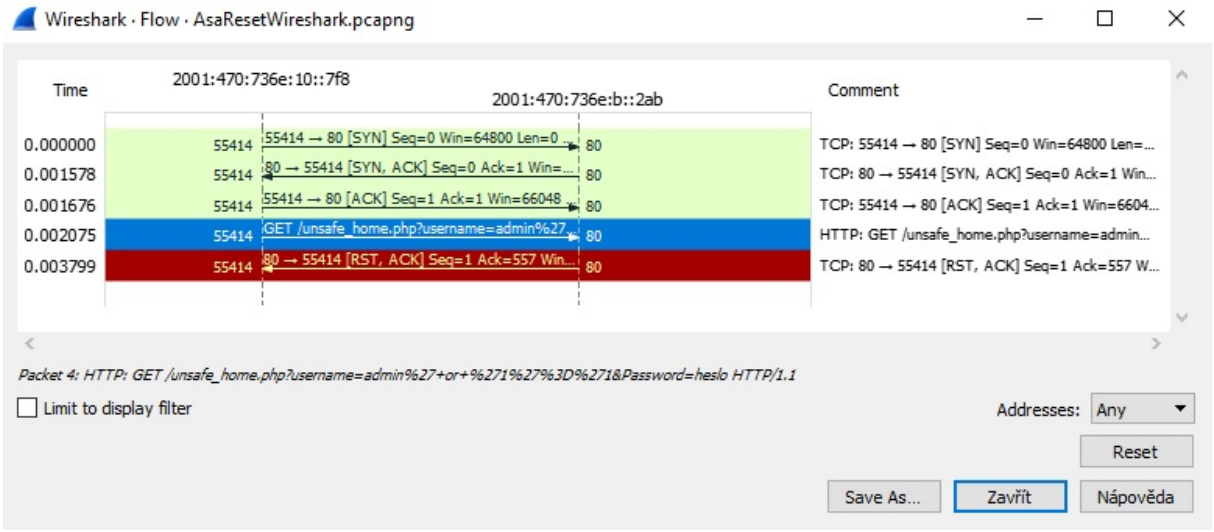
Na následujícím Obrázku: 20 je vidět schéma komunikace pro posloupnost otevření stránky 3-cestné navázání tcp, reálné administrátorské přihlášení, načtení obsahu, odhlášení. Oproti tomu Obrázek: 21 znázorňuje úvodní 3-cestné sestavení tcp, poté pokus o použití SQL injekce, načež firewall reaguje odesláním TCP-RST, ACK a komunikace se okamžitě ukončí, klient se nesnaží dále komunikovat. Při útoku firewall vygeneruje tři syslog zprávy viz.: Výpis 10.



Obrázek 19: Výsledek klasifikačních pravidel.



Obrázek 20: Schéma korektního přihlášení.



Obrázek 21: Schéma pokusu o útok ASA firewall.

```
Message: HTTP - matched request uri regex class SQL_INJECTION in
    policy-map SQL_Inject, URI matched - Resetting connection from
    outside:2001:470:736e:10::7f8/53102 to inside:2001:470:736e:b::2ab
    /80

Message: tcp flow from outside:2001:470:736e:10::7f8/53102 to inside
    :2001:470:736e:b::2ab/80 terminated by inspection engine, reason -
     reset unconditionally.

Message: Teardown TCP connection 62010 for outside:2001:470:736e
    :10::7f8/53102 to inside:2001:470:736e:b::2ab/80 duration 0:00:00
    bytes 0 Flow closed by inspection
```

Výpis 10: Syslog zprávy generované při útoku

Zátěžový test byl testován pomocí multiplatformního open-source benchmark nástroje bombardier. Zdrojové kódy a návod k použití je k dispozici na webu github viz.: [19]. Software je psaný v programovacím jazyce Go a vyžívá knihovnu fasthhtp a tím dosahuje vysokého výkonu oproti výchozí HTTP knihovně jazyku Go. Nástroj je tedy možno samostatně zkompilovat nebo stáhnout již zkompilovaný spustitelný .exe soubor. Odkaz naleznete v referencích: [20].

Při testech byla požita kompilovaná verze software na operačním systému Windows 10. Prvním krokem je stažení a instalace kompilátoru pro jazyk Go viz.: [21]. Druhým požadavkem pro instalaci je software git pro stažení repozitářů. Odkaz na Git instalátor pro systém Windows naleznete: [22]. Instalace kompilované verze poté probíhá pomocí příkazu:

```
go get -u github.com/codesenberg/bombardier
```

Výkonnostní test poté spustíme pomocí následující syntaxe:

```
bombardier -n [počet pokusů] -c [počet paralelních pokusů] [URL
    testovaného webu]
```

Výpis 11: Syntaxe spuštění výkonnostního testu

Prvním testem byl testován výkon webového serveru a zatížení firewallu při vypnuté funkci hluboké inspekce. Pouze v základní konfiguraci s pravidly provozu bez hluboké inspekce. Test byl proveden ve 3. fázích. Graf vytížení procesoru vzhledem k navázaným spojením je poté znázorněn v Obrázku.: 22. V první fázi viz. Výpis:12 bylo vygenerováno celkem 10000 dotazů pomocí 300 paralelních pokusů. Jako cíl byl domovský adresář SEED LAB webu.

```
C:\Users>bombardier -n 10000 -c 300 http://inject-asa.stibicweb.cz/
Bombarding http://inject-asa.stibicweb.cz:80/ with 10000 request(s)
    using 300 connection(s)
```

```
 10000 / 10000 [==============] 100.00% 15s Done!
Statistics          Avg        Stdev          Max
  Reqs/sec        660.63       552.39      5500.28
  Latency       337.42ms        1.57s       15.42s
  HTTP codes:
    1xx - 0, 2xx - 10000, 3xx - 0, 4xx - 0, 5xx - 0, others - 0
  Throughput:      1.46MB/s
```

Výpis 12: Výkon webu fáze 1.

V druhé fázi viz.: Výpis 13 byl testován výkon webu pro přihlášení administrátorského účtu. Odezva je tedy zpožděna procesem SQL dotazu na ověření uživatele a výpis obsahu tabulky uživatelů. Maximální počet zpracovaných dotazů za sekundu, společně s výslednou datovou propustností se tímto procesem snížili na polovinu. Průměrná i maximální doba odezvy se zvýšila dvojnásobně.

```
C:\Users>bombardier -n 10000 -c 300 http://inject-asa.stibicweb.cz/
    unsafe_home.php?username=admin&Password=seedadmin
Bombarding http://inject-asa.stibicweb.cz:80/unsafe_home.php?username
    =admin with 10000 request(s) using 300 connection(s)
 10000 / 10000[==============] 100.00% 34s Done!
Statistics          Avg        Stdev          Max
  Reqs/sec        294.95       236.47      2274.10
  Latency       663.80ms        2.73s       34.19s
  HTTP codes:
    1xx - 0, 2xx - 9912, 3xx - 0, 4xx - 0, 5xx - 0, others - 88
  Throughput:    594.15KB/s
```

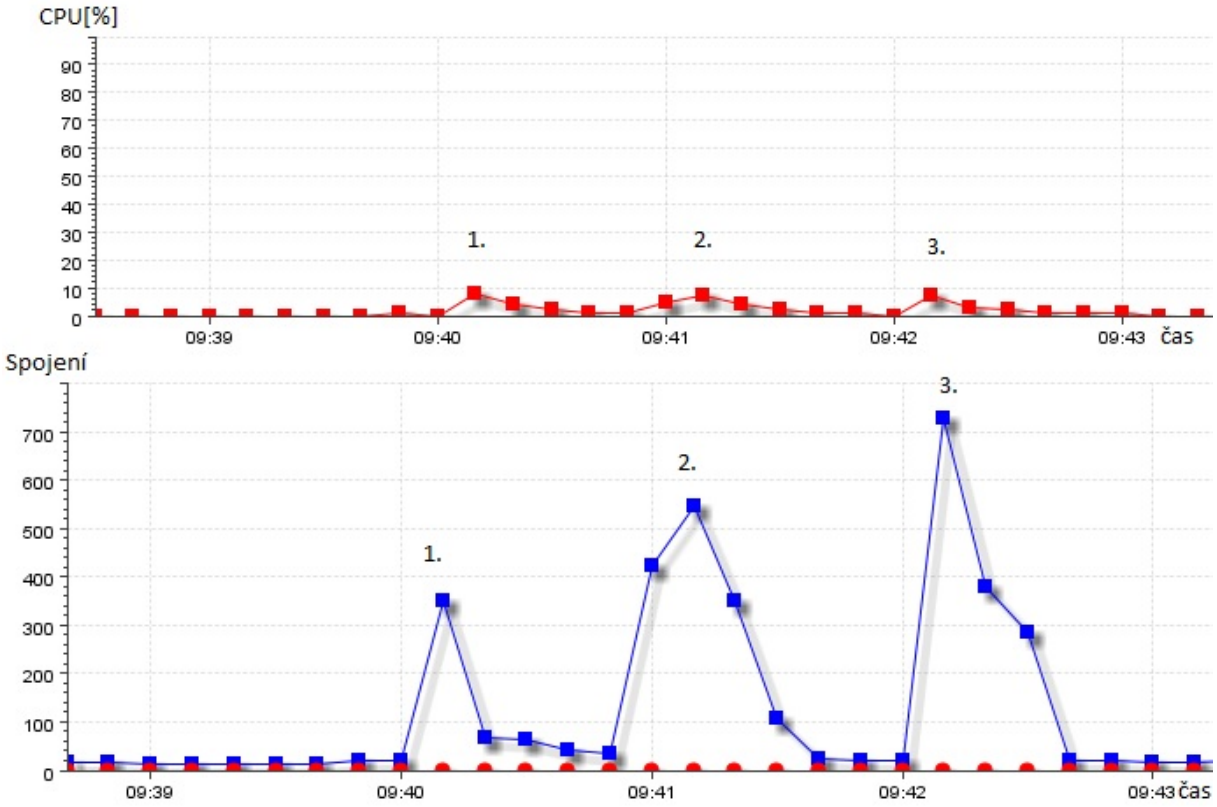
Výpis 13: Výkon webu fáze 2.

Třetí fáze testování viz.: Výpis:14 testuje rychlost odezvy domovského adresáře při zvýšení počtu paralelních spojení na dvojnásobek. Datový tok odpovědí serveru byl zvýšen o 63%. Všechny tři pokusy měly stejný vliv na zatížení procesoru firewallu, firewall dosahoval maximálního zatížení 7%. Úzkým hrdlem topologie je tedy výkon webového serveru. Sumarizovanou statistiku výkonnostního testu naleznete v Tabulce:4

```
C:\Users>bombardier -n 10000 -c 600 http://inject-asa.stibicweb.cz/
Bombarding http://inject-asa.stibicweb.cz:80/ with 10000 request(s)
    using 600 connection(s)
 10000 / 10000 [==============] 100.00% 23s
Done!
Statistics          Avg        Stdev          Max
```

```
Reqs/sec        430.30      790.78     4078.81
Latency        848.95ms      2.99s      23.27s
HTTP codes:
  1xx - 0, 2xx - 9753, 3xx - 0, 4xx - 0, 5xx - 0, others - 247
Throughput:      0.94MB/s
```

Výpis 14: Výkon webu fáze 3.



Obrázek 22: Zátěžový test bez inspekce provozu.

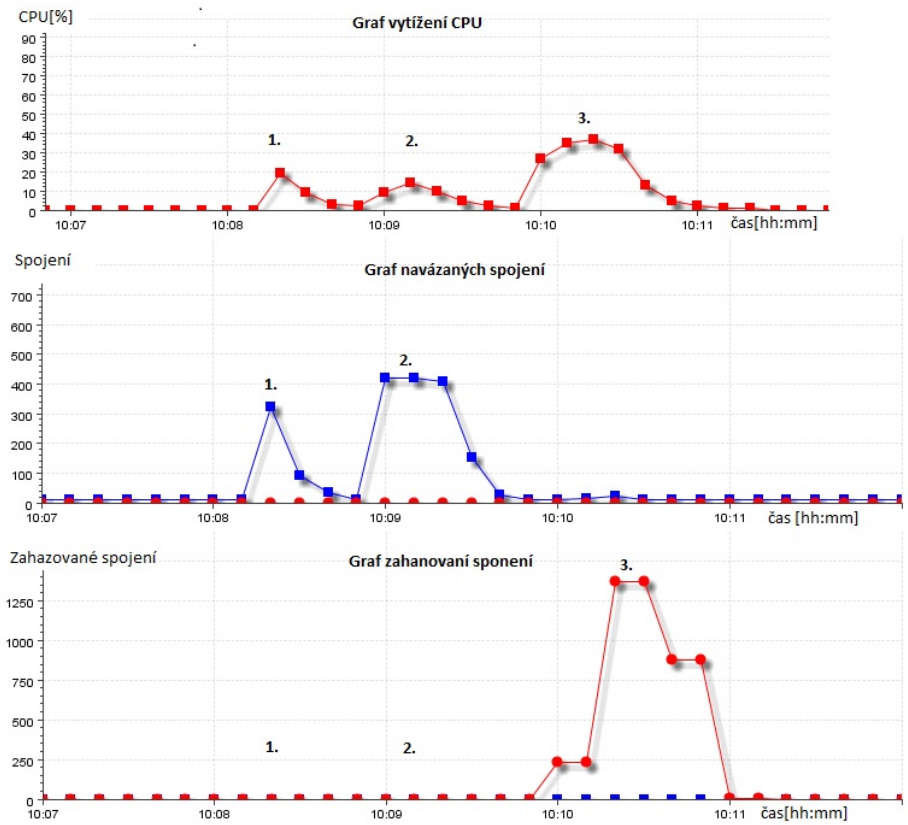
Fáze	1	2	3
Paralelní spoj.	300	300	600
Avg dotazy/s	661	295	430
Avg dezva	337 ms	663 ms	848 ms
Max dotazy/s	5500	2274	4079
Max odezva	15.42 s	34.19 s	23.27 s
Propustnost	1.46MB/s	0.59MB/s	0.94 MB/s
CPU ASA	7%	7%	7%

Tabulka 4: Výkonnostní statistika bez hluboké inspekce.

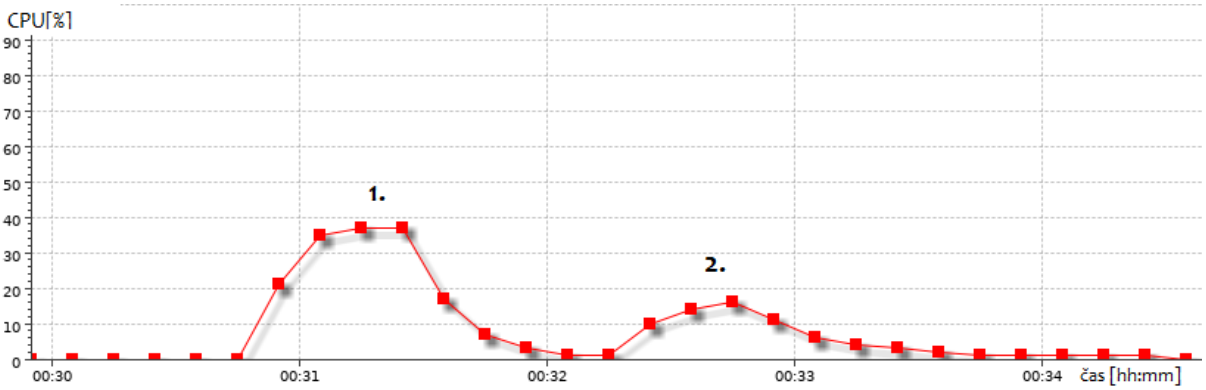
Z faktu, že zvýšení počtu paralelních spojení z jednoho zařízení nemá vysoký vliv na konečné zatížení, byla poslední fáze nahrazena testováním reakce firewallu na data obsahující SQL injekci. Výsledný graf vidíme na Obrázku: 23. Využití CPU je mnohonásobně vyšší. Vysoký podíl na zatížení má akce REJECT, jelikož firewall musí generovat odpověď na každý příchozí paket obsahující příslušnou signaturu. Graf rozdílného zatížení můžeme vidět na Obrázku: 24. První test v grafu reprezentuje využití REJECT, druhý test poté reprezentuje využití akce DROP.

Fáze	1	2	3
Paralelní spoj.	300	300	300
Avg dotazy/s	431	243	100
Avg dezva	308 ms	737 ms	2.2 s
Max dotazy/s	4652	4002	2301
Max odezva	23.18 s	41.36 s	30.05 s
Propustnost	0.97MB/s	0.49MB/s	x
CPU ASA	19%	15%	36%

Tabulka 5: Výkonnostní statistika s hlubokou inspekcí.



Obrázek 23: Zátěž při hloubkové inspekci provozu.

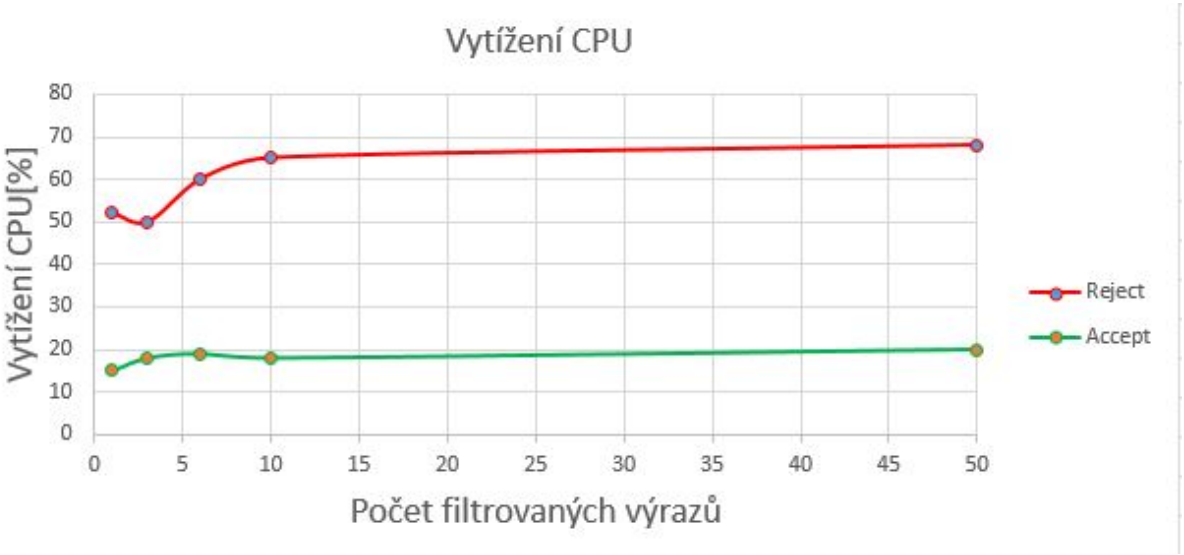


Obrázek 24: Rozdíl vytížení při užití Reject vs Drop.

Dalším testovaným parametrem byl vliv zvyšujícího se počtu regulárních výrazů na zatížení firewallu pří útoku oproti oprávněným dotazům. Pro test chování s velkým množstvím unikátních řetězců byl vytvořen pomocný skript (příloha:E) generující textový soubor obsahující příkazy pro vytvoření 2000 náhodných regulérních výrazů o délce 20 znaků, soubor s příkazy pro přidání výrazů do class-map, soubor s příkazy pro odstranění výrazů z class-mapy a soubor pro odstranění objektů z firewallu. Skript je spustitelný na Linuxu, vygenerovaná konfigurace musí být vložena ručně v příslušných konfiguračních módech.

Test regulérního provozu proběhl stejně jako vyobrazuje Výpis:12. Pro simulaci útoku bylo URL nahrazeno: `http://inject-asa.stibicweb.cz/unsafe_home.php?username=admin%27%29+or+%28%271%27%3D%271%27%2F*&Password=test`

Přidání prvních 10 regulárních výrazů mělo strmý náběh využití CPU. Tyto výrazy jsou navrhnuty pro vyhledání shody s větším množstvím SQL injekčních řetězců. Vložené automaticky vygenerované řetězce, udržely firewall při stejném zatížení i v počtu 2000. Vyšší počet výrazů není možno přidat kvůli paměťovému omezení firewallu. Výseč náběžné hrany zatížení viz.: Obrázek: 25



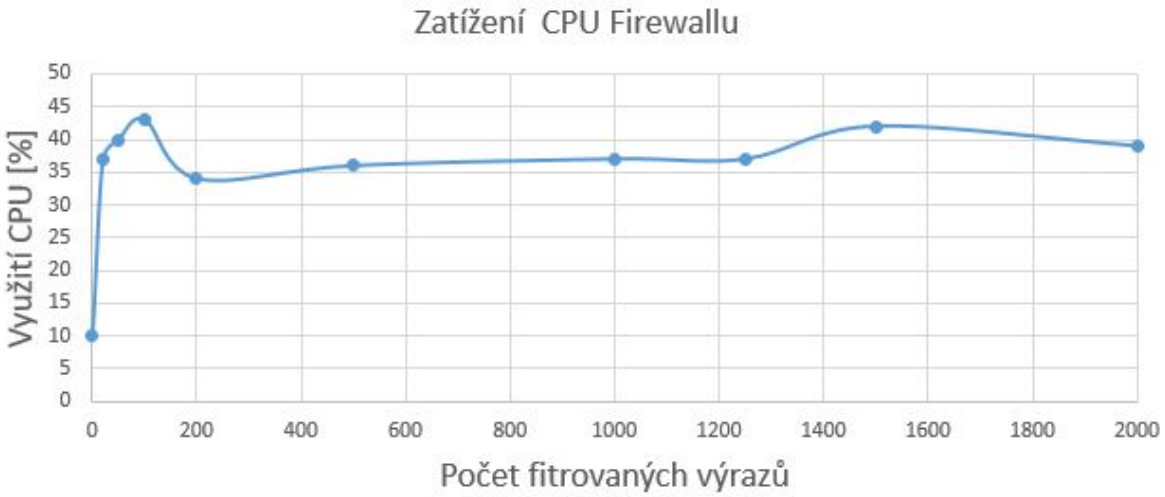
Obrázek 25: Graf: Zatížení ASA útok vs regulérní provoz.

Současně s postupným navyšováním pravidel do počtu 2000 byl testován také vliv počtu řetězců na celkovou propustnost HTTP provozu srze firewall. Testování probíhalo stahováním souboru velikosti 1GB z webového serveru programem wget.

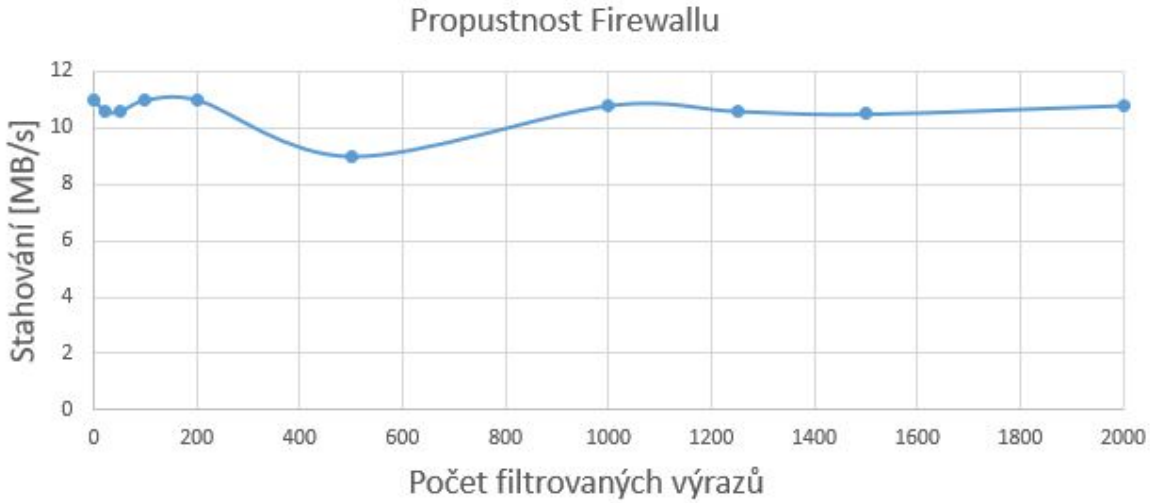
Program wget pro systém Windows byl stažen z webu `https://eternallybored.org/misc/wget/` Wget se spouští v příkazové řádce s cestou k souboru .exe staženého v předešlém odkazu. V případě stažení .exe souboru na plochu vypadá spuštění stahování takto:

```
C:\Users\Tom\Desktop>wget http://[2001:470:736e:b::4]/test.img
```

Chování CPU je souhlasné s prvním zátěžovým testem, na Obrázku: 26 můžeme vidět rychlý nástup vytížení, které se ovšem drží na přibližně stejné úrovni až do 2000 řetězců. Dle grafu propustnosti viz.: Obrázek:27 je možné tvrdit, že přidávání pravidel nemělo vliv na přenosovou rychlost v tomto dostupném přenosovém pásmu, jelikož úzkým hrdlem byly síťové karty koncových stanic. Jak karta serveru, tak karta testovací stanice poskytuje pouze 100Mbps.



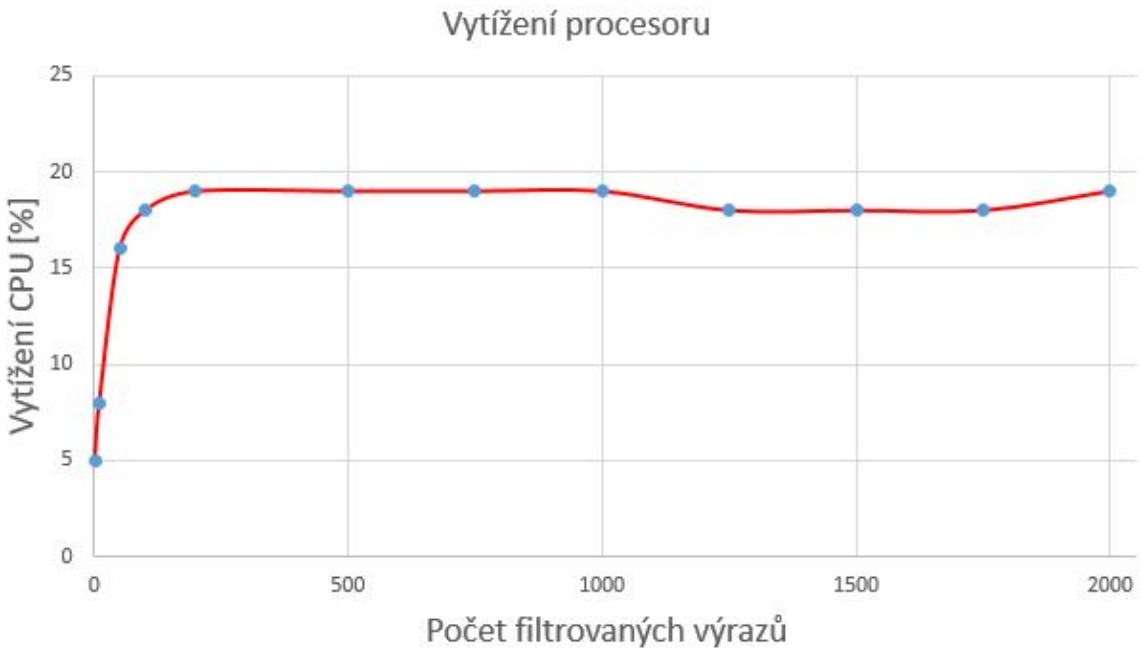
Obrázek 26: Zátěž ASA firewallu při stahování souboru přes HTTP.



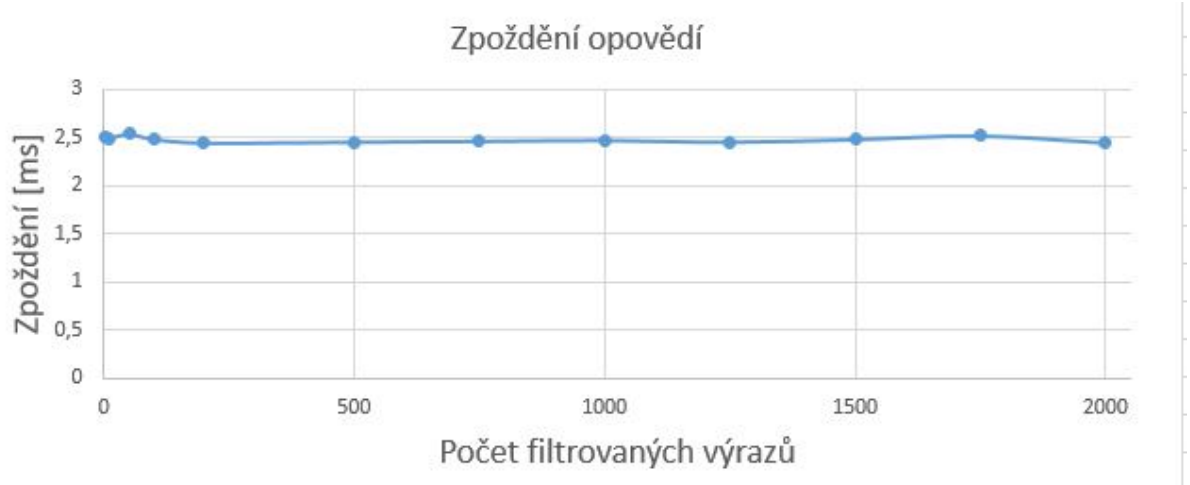
Obrázek 27: Propustnost ASA firewallu při stahování souboru přes HTTP.

Další měření softwarem bombardier, bylo zaměřeno na vliv firewallu na přenosové parametry, při provozu dimenzovaném k výkonu webového serveru. Opakován byl stejný scénář zvyšování regulárních výrazů. Software vykonával 5000 pokusů načtení domovské stránky webu prováděných bez paralelních pokusů. Sledovanými parametry byl provoz v paketech za sekundu, vytížení CPU firewallu, průměrný počet HTTP dotazů za sekundu, zpoždění odezvy webu, přenosová

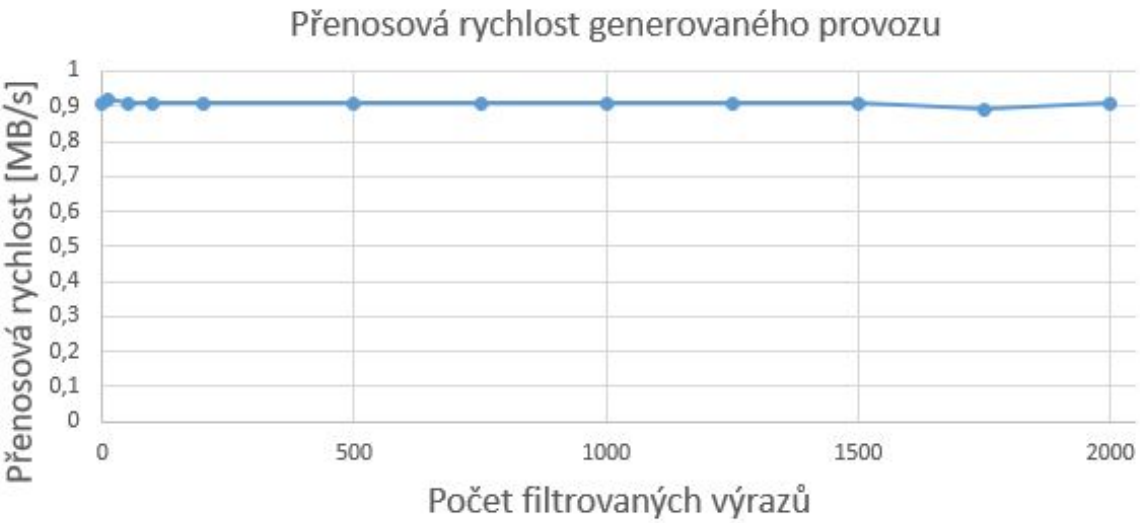
rychlost vygenerovaného provozu. Naměřené hodnoty se nacházejí v Tabulce:6, viz.:Příloha G. Graf na Obrázku: 28 vyobrazuje strmý nárust vytížení CPU do úrovně 18%, kdy již zůstává vytížení nezávislé na zvyšovaní vyhledávaných řetězců. Grafy na Obrázku: 29 a 30 ukázaly, že firewall svou kontrolou nepřidává do přenosu žádné zpoždění ani neomezuje provoz na přenosové rychlosti.



Obrázek 28: Vytížení procesoru ASA firewallu při testu Bombardier.



Obrázek 29: Zpoždění přenosu skrze ASA firewall při testu Bombardier.



Obrázek 30: Přenosová rychlost skrze ASA Firewall při testu Bombardier.

5.3 ASA aplikační firewall protokolu FTP

Filtrace na aplikační úrovni probíhá pomocí inspekční mapy pro protokol FTP. Inspekční mapa pro FTP umožňuje kontrolu dat v následujících kritériích shody:

- FTP command – syntaxe vytvoření mapy pomocí příkazu:

```
ciscoasa(config-pmap)# match [not] request-command cmd1 [cmd2...]
```

  Kde cmd-n může nabývat těchto FTP příkazů:

  - APPE — Připojit k souboru.
  - CDUP — Změní rodičovský adresář aktuálního pracovního adresáře.
  - DELE — Odstraní soubor na serveru.
  - GET — Získá soubor ze serveru.
  - HELP — Poskytuje informace nápovědy.
  - MKD — Vytvoří adresář na serveru.
  - PUT — Odešle soubor na server.
  - RMD — Odstraní adresář na serveru.
  - RNFR — Určuje název souboru „Přejmenování z“.
  - RNTO — Určuje název souboru „přejmenovat na“.
  - SITE — Slouží k zadání příkazu specifického pro server. Obvykle se používá pro vzdálenou správu.
  - STOU — Uloží soubor pomocí jedinečného názvu souboru.

  Firewall nepodporuje definici blokace dalších vlastních příkazů.

- Filename - syntaxe vytvoření mapy filtrující ftp příkaz pomocí příkazu:

```
ciscoasa(config-pmap)# match [not] filename regex {regex-name |
    class regex-class-map}
```

- File type – syntaxe vytvoření mapy filtrující typ souboru pomocí příkazu:

```
ciscoasa(config-pmap)# match [not] filetype regex {regex-name |
    class regex-class-map}
```

- Servername – syntaxe vytvoření mapy filtrující filename pomocí příkazu:

```
ciscoasa(config-pmap)# match [not] server regex {regex-name |
    class regexclass-map}
```

- Username – syntaxe vytvoření mapy filtrující uživatele pomocí příkazu:

```
ciscoasa(config-pmap)# match [not] username regex {regex-name |
    class regex-class-map
```

CLI konfigurace – blokace kopírování multimediálních souborů

Definice regulérních výrazů pro nejpoužívanější mediální soubory .mp3, .mp4 ,.avi, .acc, .waw, .flac, .wmv, .mkv pomocí CLI příkazů:

```
ASAv6(config)# regex mp3file ".*\.[Mm][Pp]3"
ASAv6(config)# regex mp4file ".*\.[Mm][Pp]4"
ASAv6(config)# regex avifile ".*\.[Aa][Vv][Ii]"
ASAv6(config)# regex accfile ".*\.[Aa][Cc][Cc]"
ASAv6(config)# regex wawfile ".*\.[Ww][Aa][Ww]"
ASAv6(config)# regex flacfile ".*\.[Ff][Ll][Aa][Cc]"
ASAv6(config)# regex wmvfile ".*\.[Ww][Mm][Vv]"
ASAv6(config)# regex mkvfile ".*\.[Mm][Kk][Vv]"
```

Výpis 15: Vytvoření regulérních výrazů.

Inspekční mapa funguje principem logických součtů objektů, jejichž kritéria shody se dají použít v kladném či negovaném tvaru. Jinak řečeno akce bude vykonána, pokud je daný výraz nalezen, nebo pokud není daný výraz nenalezn. Pokud je nutné hledat shodu s více než jedním výrazem, je třeba spojit tyto výrazy do skupin pod určitým typem class-map. Typ class-map je určen jejich účelem. Např. seskupení objektů regulárních výrazů tvoří class-map typu regex, v praktické ukázce tento typ class-map se využívá k seskupení regulárních výrazů mediálních souborů viz.: Výpis:16.

```
ASAv6(config)# class-map type regex match-any MEDIA_FILES
ASAv6(config-cmap)# match regex mp3file
ASAv6(config-cmap)# match regex mp4file
ASAv6(config-cmap)# match regex avifile
ASAv6(config-cmap)# match regex accfile
ASAv6(config-cmap)# match regex wawfile
ASAv6(config-cmap)# match regex flacfile
ASAv6(config-cmap)# match regex wmvfile
ASAv6(config-cmap)# match regex mkvfile
```

Výpis 16: Seskupení regulérních výrazů pod class-map

Jelikož je třeba kontrola parametru filename současně s parametrem request-command, je nezbytné tuto kontrolu spojit také pod jeden objekt – další class-map typu „inspect ftp“ viz.: Výpis: 17, class-map této úrovně dovoluje definovat mezi objekty operator buď logického součinu

nebo logického součtu. V případě součinu musí být splněny všechny podmínky současně, u logického součtu stačí pro vykonání akce splnění pouze jedné. Výběr logické operace OR a AND probíhá pomocí klíčového slova „match-any“ a „match-all“.

```
ASAv6(config)#class-map type inspect ftp match-all BlockUpload
ASAv6(config-cmap)# match filename regex class MEDIA_FILES
ASAv6(config-cmap)# match request-command put stou
```

Výpis 17: Spojení sou

Další krok je nastavení inspekční mapy, která shrnuje chování všech definovaných objektů pro definovaný protokol.

```
ASAv6(config)#policy-map type inspect ftp FTP_INSPECT
ASAv6(config-pmap)# parameters
ASAv6(config-pmap-p)# mask-banner
ASAv6(config-pmap-p)# mask-syst-reply
ASAv6(config-pmap-p)# quit
ASAv6(config-pmap)# class BlockUpload
ASAv6(config-pmap-c)reset log
```

Výpis 18: Vytvoření inspekční mapy.

Dalším krokem je aplikace inspekční mapy na příchozí „service policy“: Znovu je třeba vytvořit „class-map“ bez definovaného typu, jenž se použije k identifikaci provozu, na který se bude vytvořená inspekční mapa aplikovat viz.: Výpis: 19, kde je definována shoda pouze pro provoz tcp na portu 21.

```
ASAv6(config)# class-map FTP-class
ASAv6(config-cmap)# match port tcp eq 21
```

Výpis 19: Vytvoření inspekční mapy.

Posledním krokem je aplikace vytvořené „FTP-class“ na „outside“ rohraní, a spuštění inspekce ftp pomocí inspekční mapy FTP_INSPECT:

```
ASAv6(config)# policy-map outside-policy
ASAv6(config-pmap)# class FTP-class
ASAv6(config-pmap-c)# inspect ftp strict FTP_INSPECT
```

Výpis 20: Aplikace inspekční mapy.

Testování proběhlo pomocí ftp klienta pro windows – filezilla. Výstup z logu o přenosu viz.:Výpis:21

```
Stav: Připojování k [2001:470:736e:b::4]:21...
Stav: Připojení navázáno, čekání na uvítací zprávu...
```

```
Stav: Přihlášen
Stav: Spouští se odesílání C:\Users\Tester\Music\David Garrett -
    Kashmir.mp3 na server
Stav: Načítání výpisu složky /home/tom/upload...
Příkaz: TYPE I
Odpověď:  200 Switching to Binary mode.
Příkaz: EPSV
Odpověď:  229 Entering Extended Passive Mode (|||64039|)
Příkaz: LIST
Odpověď:  150 Here comes the directory listing.
Odpověď:  226 Directory send OK.
Příkaz: EPSV
Odpověď:  229 Entering Extended Passive Mode (|||55949|)
Příkaz: STOR David Garrett - Kashmir.mp3
Chyba:  Nelze číst ze soketu: ECONNRESET - Připojení resetováno
    druhou stranou
Chyba:  Odpojen od serveru
Chyba:  Přenos souboru se nezdařil
```

Výpis 21: Filezilla log - pokus o nahrání souboru .mp3 ASA

Na opačné straně firewall vytvoří tyto syslog zprávy:

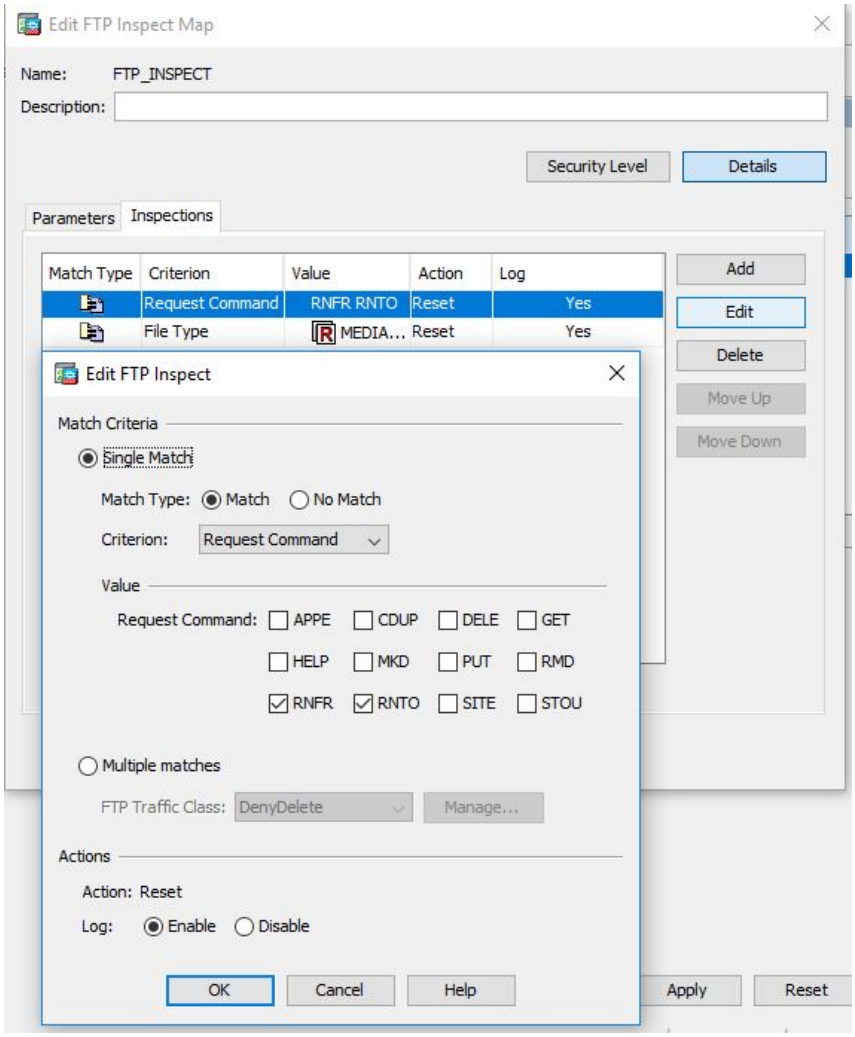
```
Message: Strict FTP inspection matched Class 29: BlockUpload in
    policy-map FTP_INSPECT, Reset connection from outside:2001:470:736
    e:10::4bd/52608 to inside:2001:470:736e:b::4/21

Message: tcp flow from outside:2001:470:736e:10::4bd/52608 to inside
    :2001:470:736e:b::4/21 terminated by inspection engine, reason -
    inspector reset unconditionally.

Message: Teardown TCP connection 6093 for outside:2001:470:736e:10::4
    bd/52608 to inside:2001:470:736e:b::4/21 duration 0:00:00 bytes
    373 Flow closed by inspection
```

Výpis 22: Firewall syslog - blokace stažení mp3

Jak ukazuje výstup 21 klient reálně neužívá pro nahrávání příkaz „PUT“ , který je používán pro nahrání souboru pomocí příkazové řádky Windows a který je definovám při v konfiguraci ASA. Příkaz PUT je pouze interpretace příkazu pro člověka, client reálně používá příkaz „STOR“, jenž je definován v RFC959: FILE TRANSFER FUNCTIONS.

ASDM konfigurace – blokace přejmenovávání souborů

Pro přidávání dalších blokací FTP inspekcí stačí editovat existující inspekční mapu FTP_INSPECT. V ASDM nalezneme v menu: Configuration > Firewall > Objects > Inspect Maps > FTP > FTP_INSPECT > Customize... > Details > záložka „Inspections“. V vybrané záložce nalezneme veškerá nakonfigurovaná pravidla shody zvolené inspekční mapy. Tlačítkem „ADD“ přidáme nové pravidlo s kritériem shody: Request Command viz.: Obrázek: 31 Hodnotu příkazu vybereme z představených možností RNFR(rename from), RNTO (rename to).



Obrázek 31: ASDM blokace přejmenování.

Průběh testu přejmenování souboru pomocí klienta filezila:

```
tav: Připojování k [2001:470:736e:b::4]:21...
Stav: Připojení navázáno, čekání na uvítací zprávu...
Stav: Server nepodporuje znaky, které nepatří do ASCII.
Stav: Přihlášen
Stav: Přejmenování /home/tom/Dokumenty/FTP/TestWor.docx na /home/tom/
    Dokumenty/FTP/TestWor.txt
Příkaz: CWD /home/tom/Dokumenty/FTP
Odpověď: 250 Directory successfully changed.
Příkaz: RNFR TestWor.docx
Chyba: Odpojen od serveru: ECONNABORTED - Připojení bylo přerušeno
```

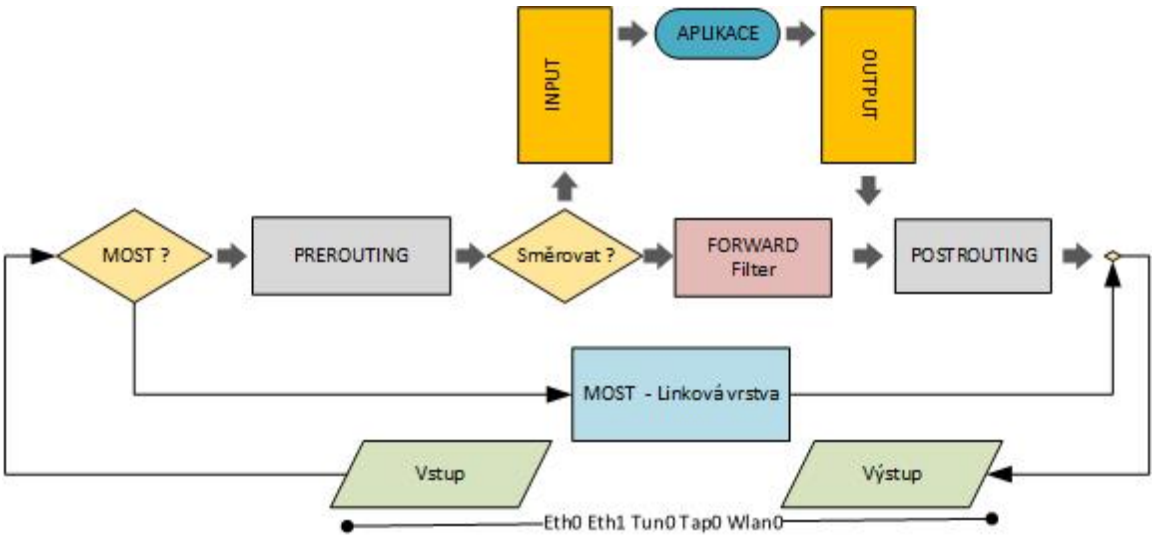
Výpis 23: Filezilla log - pokus o přejmenování souboru

```
Message: Strict FTP inspection matched request-command rnfr rnto in
    policy-map FTP_INSPECT, Reset connection from outside:2001:470:736
    e:10::7f8/52996 to inside:2001:470:736e:b::4/21

Message: tcp flow from outside:2001:470:736e:10::7f8/52996 to inside
    :2001:470:736e:b::4/21 terminated by inspection engine, reason -
    inspector reset unconditionally.

Message: Teardown TCP connection 60664 for outside:2001:470:736e
    :10::7f8/52996 to inside:2001:470:736e:b::4/21 duration 0:00:00
    bytes 165 Flow closed by inspection
```

Výpis 24: Firewall syslog - blokace o přejmenování souboru

6 Linux Firewall

Operační systém Linux má nástroj pro práci s průchodem paketů integrovaný v samotném jádře. Jedná se o netfilter, který je často mylně zaměňován s nástrojem, který slouží k manipulaci s tabulkami netfiltru - iptables/ip6tables. Konfigurace netfiltru se zakládá na definici pravidel pomocí nástroje ip6tables, na jejichž základě se netfilter rozhoduje, co se kterým paketem dělat. Každé pravidlo patří do jedné z tabulek a vztahuje se na jeden řetězec. Každá z tabulek obsahuje předdefinované řetězce(chains) odpovídající některé z fází zpracování paketu. Pokud není nalezeno vyhovující pravidlo, použije se výchozí akce pro řetězec [2]. Pakety putují netfiltrem pevně danou cestou a lze je kontrolovat v pěti různých fázích, viz.:Blokové schéma netfiltru: Obrázek: 32



Obrázek 32: Blokové schéma linuxového netfiltru.

- PREROUTING – Nově příchozí pakety, které ještě neprošly směrováním. Je u nich možné změnit cílovou ip adresu a port. Po směrování pokračují buď do INPUT nebo do FORWARD.
- INPUT – Příchozí pakety, které prošly směrováním a jsou určeny nějaké lokální službě. Lze určit rozhraní, z kterého přišly. Pokud pravidla dovolí, budou předány konkrétní aplikaci.
- OUTPUT – Lokálně vytvořené odchozí pakety, jenž ještě neprošly směrováním. Lze zjistit, kterým rozhraním budou zřejmě odeslány. Po směrování pokračují do POSTROUTING.

- FORWARD – Pakety, které prošly směrováním a jsou určeny jinému počítači. Měly by se přeposlat dál. Je známo vstupní i výstupní rozhraní. Pokud je přeposlání schváleno, pokračují do POSTROUTING. Přeposílání se musí povolit přímo v jádře.

- POSTROUTING – Pakety, které prošly směrováním a opustí počítač. Místo pro změnu odchozí IP adresy.

IPtables pracuje se třemi základními tabulkami:

- FILTER – Výchozí tabulka, určená k filtrování paketů. Jsou v ní přístupné řetězce INPUT, OUTPUT a FORWARD.

- NAT - Tabulka používaná pro překlad adres v řetězcích PREROUTING, POSTROUTING a méně často i v OUTPUT.

- MANGLE - V této tabulce lze měnit některé další hodnoty v IP hlavičce (TOS, TTL) a označit pakety. Značka není součástí paketu a zaniká s odchodem paketu. Pracuje se všemi pěti definovanými řetězci.

Podmínky stanovené pravidlem, pro které se hledá shoda (matches) mohou být definovány jako omezení na určitou zdrojovou nebo cílovou IP adresu, specifický port daný pro určitou aplikaci jak je známo ze všech typů firewallu. Iptables také obsahuje rozšiřující moduly, jenž nám umožňují hledat shodu také v textovém řetězci aplikačních dat. Rozšiřující pravidla je možné vytvářet pomocí ip6tables [-m jméno modulu[-j jméno cíle [možnosti cíle]. Rozšiřující pravidla je možné kombinovat a jsou vyhodnocovány v pořadí zápisu v pravidle. Jestliže jsou splněny všechny podmínky definované pravidlem, provede se s paketem operace specifikovaná v cíli(target) pravidla. [5], [6]

Mezi základní cíle pravidel patří:

- ACCEPT – Paket je schválen a pokračuje dál v cestě.

- DROP – Paket je zahozen.

- REJECT – Paket je zahozen a generuje se většinou icmp zpráva.

- JINÝ ŘETĚZEC – Zpracování paketu se předá jinému řetězci.

- LOG – Vytvoří záznam o paketu v syslogu.

- RETURN – Vrátí se do předchozího řetězce a pokračuje následujícím pravidlem nebo pokud není kam se vrátit, postupuje se podle výchozího pravidla pro řetězec.

- DNAT – Mění cílovou adresu nebo port paketu.

- SNAT – Mění zdrojovou adresu nebo port paketu.

• MASQUERADE – Podobný SNAT, ale adresa se zjišťuje ze zadaného rozhraní.

• REDIRECT – Změní cílovou adresu na svoji, případně změní i cílový port.

6.1 Aplikační firewall v Linuxu

Implementace firewallu na platformě Linux spočívá na použití nástroje netfilter, který je již integrován v linuxovém kernelu. Pravidla samotná se vytváří pomocí nástroje ip6tables. Ip6tables poskytuje mnoho přídavných modulů, jenž rozšiřují možnosti prohledávání provozu než pouze jen na zdrojovou, cílovou adresu a port služby. Pro potřeby aplikačního firewallu byl využit modul „string“. Modul umožňuje prohledávat data a hledat shodu testového řetězce nebo řetězce hexadecimálních dat. Pomocí parametru –algo je možné definovat metodu prohledávání pomocí dvou algoritmů. Na výběr máme algoritmy Boyer-Moore a Knuth-Pratt-Morris [2]. Modulů je možné v jednom pravidle řetězit až 300. To nám dává možnost vyhledávat v provozu vícenásobnou shodu. Příkladné využití vícenásobné detekce může být detekce Bleeding-Edge exploitu proti MS-SQL, kdy se útočník snaží ukončit sekci sql dotazu dvojicí znaků komentáře společně se znakem NULL po každém z nich.

```
[iptablesfw]# ip6tables -I FORWARD 1 -p tcp --dport 1433 -m state --
    state
ESTABLISHED -m string --hex-string "'|00|" --algo bm -m string --hex-
    string
"-|00|-|00|" --algo bm -j LOG --log-prefix "SQL INJECTION COMMENT"
```

Výpis 25: Překlad Fwsnort sid: 2000488 rev:5

Pokud ale TCP spojení obsahuje dva řetězce v opačném pořadí například - | 00 | - | 00 | foo '| 00 | místo '| 00 | foo - | 00 | - | 00 |, jak originální snort signatura tak tento iptables překlad budou klamně pozitivní.[2]

Princip vyhledávacích algoritmů:

Boyer-Moore algoritmus funguje na principu hledání schody od konce porovnávaného řetězce a předpokládá, že pokud není shoda na konci, není shoda ani na začátku. To umožňuje velké skoky v prohledávaném textu, šetří čas a výpočetní výkon při porovnávání textu připomínající přirozený jazyk.

Knuth-Pratt-Morris algoritmus vychází z myšlenky, že je zbytečné vracet se při neúspěšném porovnaní v řetězci zpět a porovnávat znova znaky, které už porovnané byli. Aby nebylo nutné se vracet zpět, je potřeba vytvořit před začátkem vyhledávání tabulku, do které se bude ukládat pro každou pozici ve vzoru číslo, jež nám bude určovat, který prvek vzoru se má porovnávat s aktuálním znakem v řetězci.

Výhodou Linux distribuce při implementaci aplikačního firewallu je možnost využití databáze nejsilnějšího open-source IPS systému Snort. Databáze Snort je největší známý zdroj signatur

rozpoznávající většinu známých útoků na aplikační úrovni. Informace o jednotlivých signaturách útoků je možno snadno vyhledávat na stránkách projektu Snort dle klíčových slov, zasažených protokolů, či čísla exploitu viz.: [4]. Pokud známe ID dané signatury, je možné využít open-source pearl skript:fwsnort viz.: [7]. Tento skript dokáže přeložit až 65% doposud známých Snort signatur na pravidla syntaxe iptables/ ip6tables. Z důvodu licenční limitace funkce Snort na straně firewallu ASA, nebude v práci tato funkce implementována ani pro aplikační firewall v GNU/Linux.

6.2 Linux aplikační firewall proti SQL injekci

Při implementaci aplikačního firewallu na Linuxu není porovnávána hodnota předem známého pole v aplikačních datech, ale zvoleným algoritmem jsou procházena aplikační data na všech úrovních ISO/OSI a je hledána shoda s definovaným řetězcem. Je doporučeno prohledávání co nejvíce zúžit. Minimálně by mělo být zaměřeno na určitý port. Ukázkové pravidlo je navrhnuto pro stejný řetězec, který byl analyzován pomocí Wiresharku v kapitole 5.1, Obrázek: 15. Tedy firewall by měl blokovat spojení s výskytem řetězce:

`admin%27+or+%271%27%3D%27`

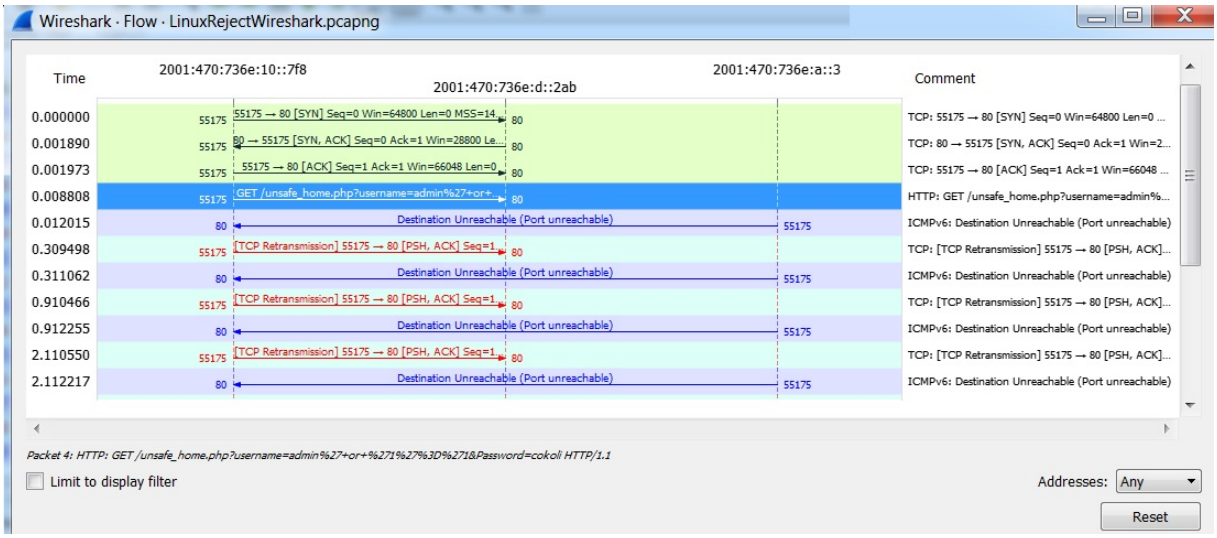
Pro každé definované pravidlo je možná pouze jedna akce, proto pokud je nutné mít zpětnou vazbu uloženou v log souboru firewallu, je třeba definovat pravidlo dvakrát. Jedna definice s akcí LOG, druhá definice s akcí REJECT/DROP. Akce reject zahodí navázané tcp spojení a vygeneruje ICMP zprávu pro klienta s informací, že daný port je na zařízení nedostupný. Akce drop zahodí dané spojení bez žádné zpětné informace pro klienta. Ani jedna z akcí korektně neukončí spojení. Klientský systém sice dostává zprávu o nedostupnosti portu, spojení však stále vyčkává na odezvu, případně několikrát opakuje daný dotaz a ověřuje, zda-li nedošlo k dočasnému výpadku. Tento fakt znemožňuje použití benchmark nástrojů, jelikož aplikace očekávají odezvu v protokolu HTTP a nehledí na ICMP zprávy přijaté systémem. Chování REJECT umožňuje parametrem –reject-with tcp-reset změnit výchozí chování na ukončení spojení pomocí RST. Užitečná funkce ip6tables je také akce LOG, parametr –log-prefix umožňuje označkovat událost námi definovaným jménem, což poté umožňuje vyhledat událost pomocí příkazu grep v souboru /var/log/kern.log viz.: Výpis:27. Celá konfigurace pro nejznámější injekční řetězce viz.:PřílohaH

```
sudo ip6tables -I FORWARD 1 -p tcp --dport 80 -m state --state
    ESTABLISHED -m string --string "admin%27+or+%271%27%3D%27" --algo
    bm -j LOG --log-prefix "SQL-injection8"

sudo ip6tables -I FORWARD 2 -p tcp --dport 80 -m state --state
    ESTABLISHED -m string --string "admin%27+or+%271%27%3D%27" --algo
    bm -j REJECT --reject-with tcp-reset
```

Výpis 26: Pravidlo blokující SQL injekci na Linuxu

```
Dec 18 13:22:22 UbuntuFirewall kernel: [ 1503.214590] SQL-
    injection8IN=ens32 OUT=ens33 MAC=80:1f:02:2f:f0:03:d8:58:d7:00:4d
    :2b:86:dd SRC=2001:0470:736e:0010:0000:0000:0000:07f8 DST
    =2001:0470:736e:000d:0000:0000:0000:02ab LEN=647 TC=0 HOPLIMIT=62
    FLOWLBL=504811 PROTO=TCP SPT=55186 DPT=80 WINDOW=258 RES=0x00 ACK
    PSH URGP=0
Dec 18 13:22:23 UbuntuFirewall kernel: [ 1504.415167] SQL-
    injection8IN=ens32 OUT=ens33 MAC=80:1f:02:2f:f0:03:d8:58:d7:00:4d
    :2b:86:dd SRC=2001:0470:736e:0010:0000:0000:0000:07f8 DST
    =2001:0470:736e:000d:0000:0000:0000:02ab LEN=647 TC=0 HOPLIMIT=62
    FLOWLBL=504811 PROTO=TCP SPT=55186 DPT=80 WINDOW=258 RES=0x00 ACK
    PSH URGP=0
```

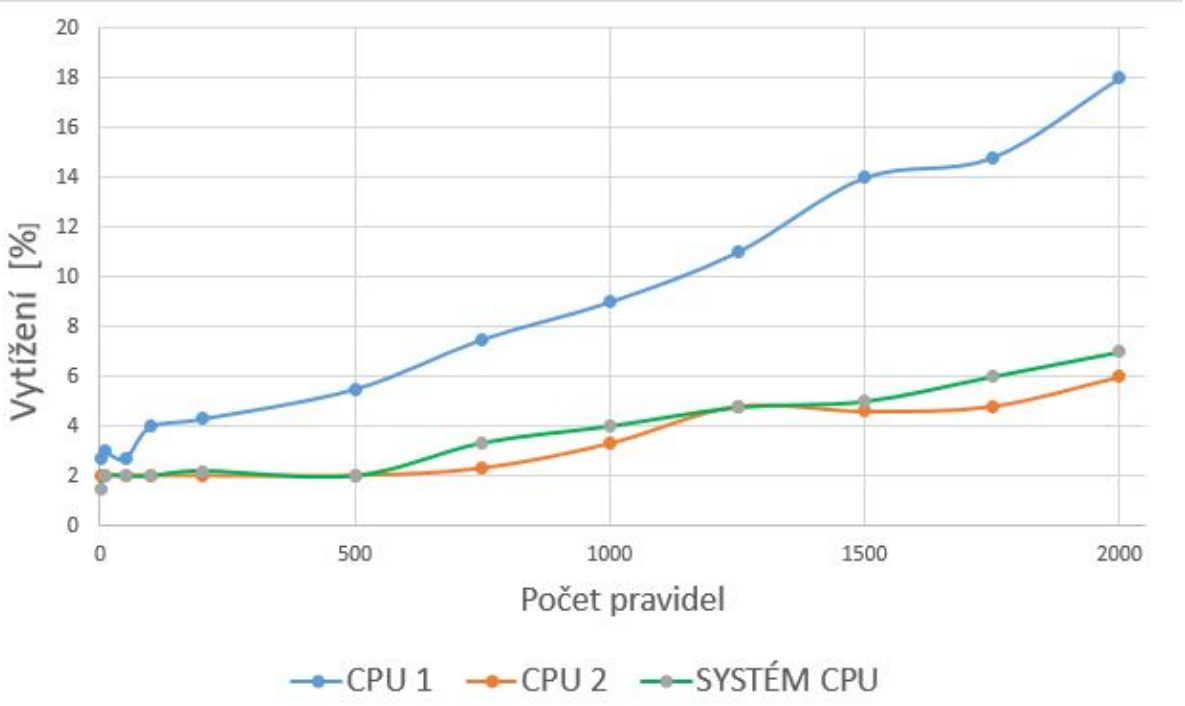
Výpis 27: Linux Syslog s průběhem útoku



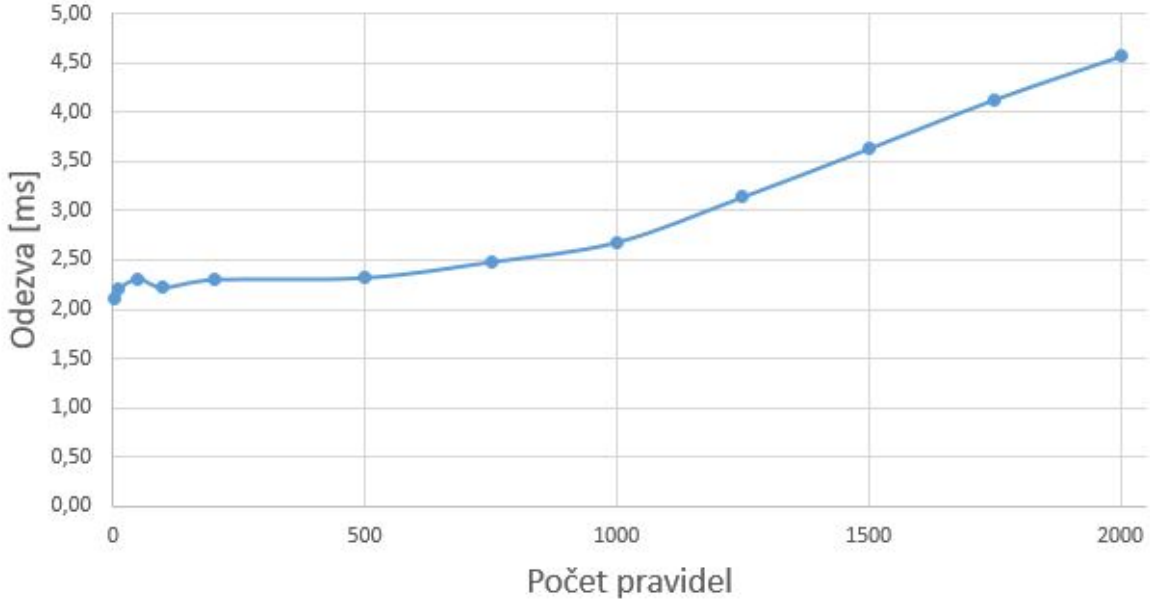
Obrázek 33: Schéma pokusu o útok Linux Firewall.

Bylo provedeno několik testů závislosti přenosových parametrů na zvyšující se počet filtrovaných řetězců. Soustava prvních testů prováděná pomocí software bombardier testovala počet přenesených paketů za sekundu, využití jednotlivých jader CPU, spotřeba CPU samotným systémem, průměrný počet odeslaných HTTP dotazů/s během testu, průměrné zpoždění odezvy na jednotlivé dotazy, přenosovou rychlost. K měření zátěže jednotlivých jader byl použit nástroj HTOP, další systémové zdroje byly měřeny nástrojem SAR. Pro měření generovaných paketů byl použit skript viz.: Příloha F. Tabulky z měření zmíněných parametrů naleznete v příloze: G, Tabulka:7, Tabulka:8. Výsledky některých parametrů jsou vykresleny v následujících grafech: Obrázek: 34 , Obrázek: 35, Obrázek: 36.

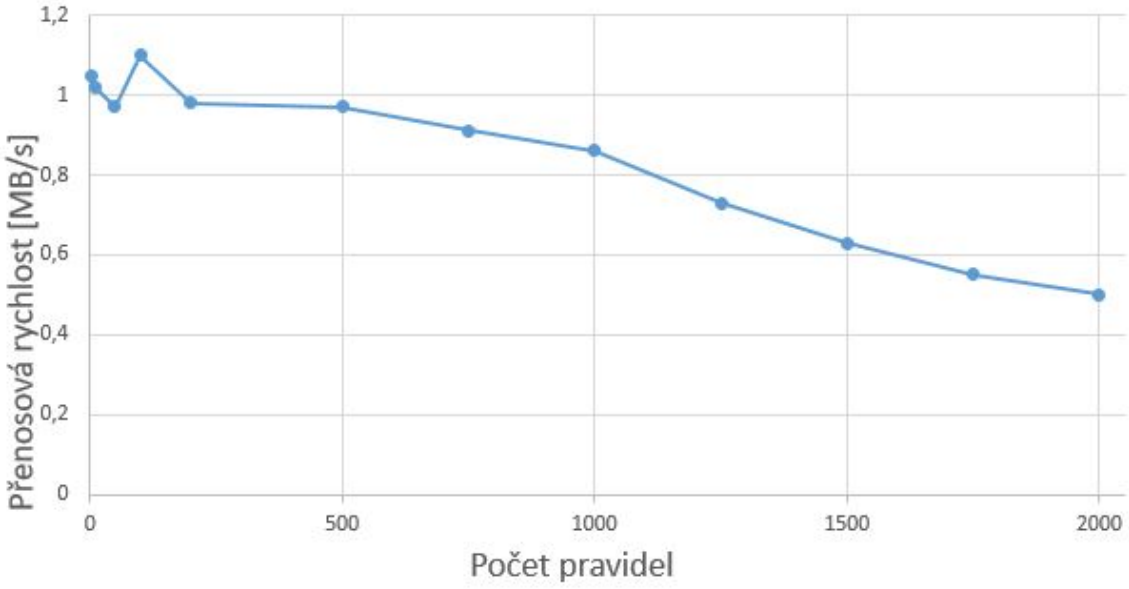
Proběhlé testování ukázalo, že přidání 2000 pravidel s hlubokou inspekcí zvyšuje odezvu serveru na dvojnásobek oproti spojení bez firewallu. Zároveň, jak ukazuje Tabulka: 8, přenosová rychlost i počet odbavených dotazů klesl až na polovinu. Další poznatek vyplývající z grafu na Obrázku: 34, provoz je zpracováván primárně jedním jádrem, využití druhého jádra se blíží téměř k stavu zdrojů spotřebovávaných systémem.



Obrázek 34: Zátěž Linux firewallu při testu průchozích spojení bombardier.

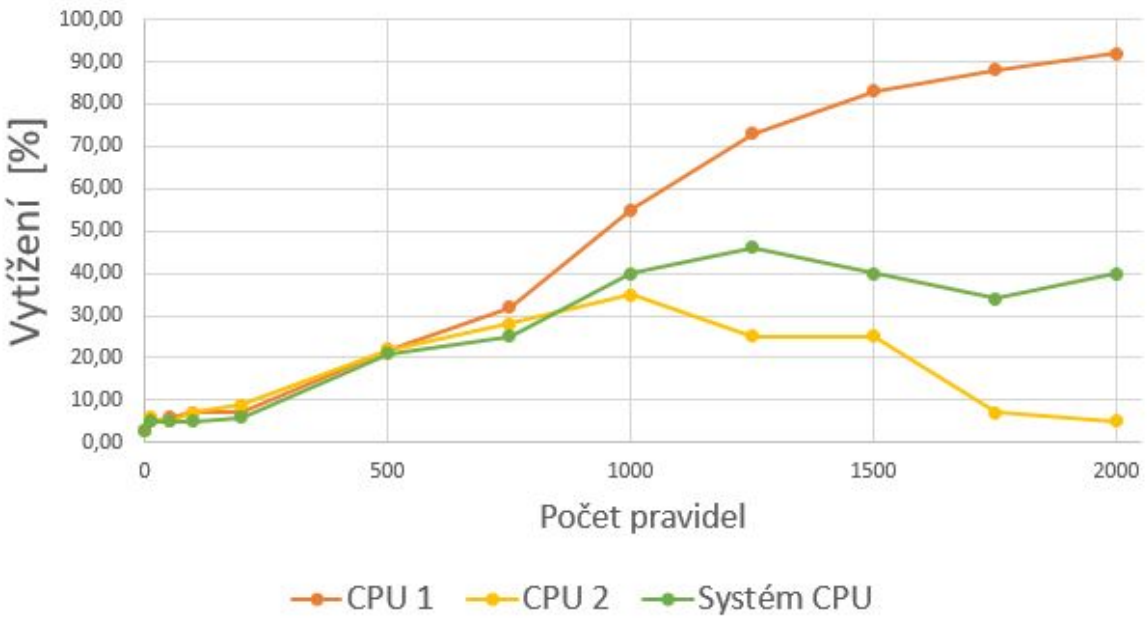


Obrázek 35: Linux zpoždění odpovědí HTTP dotazů při testu bombardier.

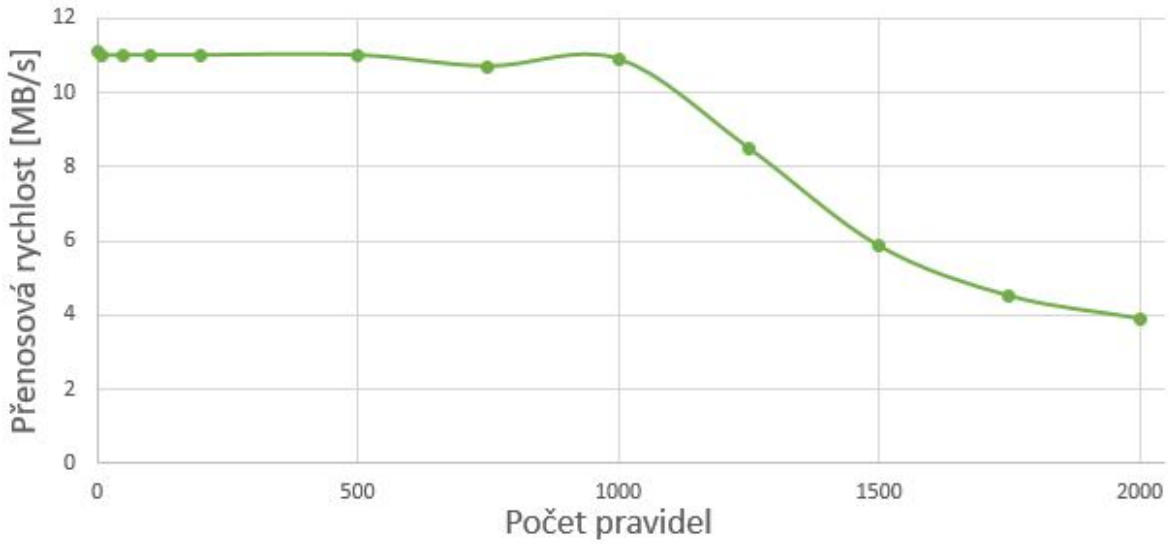


Obrázek 36: Přenosová rychlost HTTP provozu při testu bombardier.

Dále byl testován vliv zvyšování počtu pravidel na zátěž procesorů při maximální propustnosti sítě. Testování probíhalo pomocí opakovaného stahování souboru o velikosti 1 GB ze serveru apache2. Výsledky měření naleznete příloze: G ,Tabulka: 9



Obrázek 37: Zátěž Linux firewallu při stahování.



Obrázek 38: Propustnost Linux firewallu pro HTTP stahování.

6.3 Linux aplikační firewall protokolu FTP

Při implementaci pravidel pro FTP můžeme vycházet z poznatků nabytých při implementaci firewallu na Cisco ASA.

Blokace kopírování multimediálních souborů Signalizace protokolu FTP je textově orientovaná a nešifrovaná komunikace. Z analýzy komunikace víme, že pokyn pro nahrání souboru je příkaz „STOR“ , zřetězením modulu string vyhledávajícího tento řetězec s modulem vyhledávající známou koncovku multimediálního souboru se dosáhne blokace nahrávání souboru. Konkrétní blokaci zajistí akce REJECT jako cíl tohoto pravidla. Příkladová konfigurace 28 znázorňuje příkaz pro blokaci uložení souboru .mp3 , výsledný log za aplikace vidíte na Výpisu: 29. Celá konfigurace pro nejpoužívanější mediální formáty viz.:Příloha:J

```
sudo ip6tables -I FORWARD 21 -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mp3" --algo bm -m string --string
     "STOR" --algo bm  -j LOG --log-prefix "MediaBlock"

sudo ip6tables -I FORWARD 22 -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mp3" --algo bm -m string --string
     "STOR" --algo bm -j REJECT
```

Výpis 28: Blokace kopírování .MP3 na Linuxu

```
Stav: Přihlášen
Stav: Spouští se odesílání C:\Users\Tester\Music\Alan Walker ft.
    Gavin James - Tired.mp3 na server
Příkaz: CWD /home/tom/upload
Odpověď:  250 Directory successfully changed.
Příkaz: PWD
Odpověď:  257 "/home/tom/upload" is the current directory
Příkaz: TYPE I
Odpověď:  200 Switching to Binary mode.
Příkaz: EPSV
Odpověď:  229 Entering Extended Passive Mode (|||56687|)
Příkaz: STOR Alan Walker ft. Gavin James - Tired.mp3
Chyba:  Odpojen od serveru: ECONNABORTED - Připojení bylo přerušeno
Chyba:  Přenos souborů se nezdařil po přenesení 786 432 bajtů v 18
    sekund
Stav: Odpojen od serveru
```

Výpis 29: Filezilla log - pokus o nahrání .mp3 - Linux Firewall

```
Mar 13 13:32:04 UbuntuFirewall kernel: [15286.502980] MediaBlockIN=
    ens32 OUT=ens33 MAC=80:1f:02:2f:f0:03:d8:58:d7:00:4d:2b:86:dd SRC
    =2001:0470:736e:0010:0000:0000:0000:07f8 DST=2001:0470:736e:000d
    :0000:0000:0000:0004 LEN=118 TC=0 HOPLIMIT=62 FLOWLBL=692443 PROTO
    =TCP SPT=62933 DPT=21 WINDOW=257 RES=0x00 ACK PSH URGP=0
Mar 13 13:32:09 UbuntuFirewall kernel: [15291.470359] MediaBlockIN=
    ens32 OUT=ens33 MAC=80:1f:02:2f:f0:03:d8:58:d7:00:4d:2b:86:dd SRC
    =2001:0470:736e:0010:0000:0000:0000:07f8 DST=2001:0470:736e:000d
    :0000:0000:0000:0004 LEN=118 TC=0 HOPLIMIT=62 FLOWLBL=692443 PROTO
    =TCP SPT=62933 DPT=21 WINDOW=257 RES=0x00 ACK PSH URGP=0
```

Výpis 30: Linux Syslog při pokusu o nahrání .mp3

Blokace přejmenování souborů

Pro blokaci přejmenování souborů je třeba blokovat příkazy RNFR and RNTO, spojením s dalším modulem můžeme dosáhnout blokace přejmenovávání pouze specifického typu souborů.

```
sudo ip6tables -I FORWARD 37 -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string "RNFR" --algo bm -j LOG --log-
    prefix "RenameBlock"

sudo ip6tables -I FORWARD 38 -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string "RNFR" --algo bm -j REJECT

sudo ip6tables -I FORWARD 39 -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string "RNTO" --algo bm -j LOG --log-
    prefix "RenameBlock"

sudo ip6tables -I FORWARD 40 -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string "RNTO" --algo bm -j REJECT
```

Výpis 31: Blokace přejmenování na Linuxu

```
Stav: Připojování k [2001:470:736e:d::4]:21...
Stav: Připojení navázáno, čekání na uvítací zprávu...
Stav: Nezabezpečený server, nepodporuje FTP přes TLS.
Stav: Server nepodporuje znaky, které nepatří do ASCII.
Stav: Přihlášen
Stav: Přejmenování /home/tom/Dokumenty/FTP/TestWord2.docx na /home/
    tom/Dokumenty/FTP/TestWord2.txt
Příkaz: CWD /home/tom/Dokumenty/FTP
```

```
Odpověď: 250 Directory successfully changed.
Příkaz: RNFR TestWord2.docx
Chyba: Odpojen od serveru: ECONNABORTED – Připojení bylo přerušeno
```

Výpis 32: Filezilla log - pokus přejmenování souboru - Linux Firewall

```
UbuntuFirewall kernel: [15547.857419] RenameBlockIN=ens32 OUT=ens33
    MAC=80:1f:02:2f:f0:03:d8:58:d7:00:4d:2b:86:dd SRC=2001:0470:736e
    :0010:0000:0000:0000:07f8 DST=2001:0470:736e:000d
    :0000:0000:0000:0004 LEN=81 TC=0 HOPLIMIT=62 FLOWLBL=352322 PROTO=
    TCP SPT=62965 DPT=21 WINDOW=258 RES=0x00 ACK PSH URGP=0
Mar 13 13:36:26 UbuntuFirewall kernel: [15548.176242] RenameBlockIN=
    ens32 OUT=ens33 MAC=80:1f:02:2f:f0:03:d8:58:d7:00:4d:2b:86:dd SRC
    =2001:0470:736e:0010:0000:0000:0000:07f8 DST=2001:0470:736e:000d
    :0000:0000:0000:0004 LEN=81 TC=0 HOPLIMIT=62 FLOWLBL=352322 PROTO=
    TCP SPT=62965 DPT=21 WINDOW=258 RES=0x00 ACK PSH URGP=0
```

Výpis 33: Linux Syslog při pokusu o přejmenování souboru

7 Srovnání Linux a ASA firewallu

Implementace obou aplikačních firewallů je založena na stejném principu porovnávání aplikačních dat s definovaným řetězcem. Linux oproti ASA firewallu neomezuje porovnávání k předdefinovanému aplikačnímu poli přesně definovaného protokolu, ale umožňuje řetězec vyhledávat nad každým paketem a to jak v alfanumerické, tak v hexadecimální podobě. Nastavení vyhledávání řetězce v určitých aplikačních datech vyžaduje při konfiguraci ASA firewallu více kroků a obsáhlejší studium konfiguračních návodů. Oproti tomu Linux firewall se konfiguruje pomocí série jednoduchých příkazů, ve kterých je nutné pouze modifikovat či řetězit definice hledaných řetězců.

Paměťová náročnost Z paměťové náročnosti byli oba firewally na stejné úrovni, paměť RAM je alokována pouze pro potřeby systému a při použití aplikačního firewallu nejeví známky růstu.

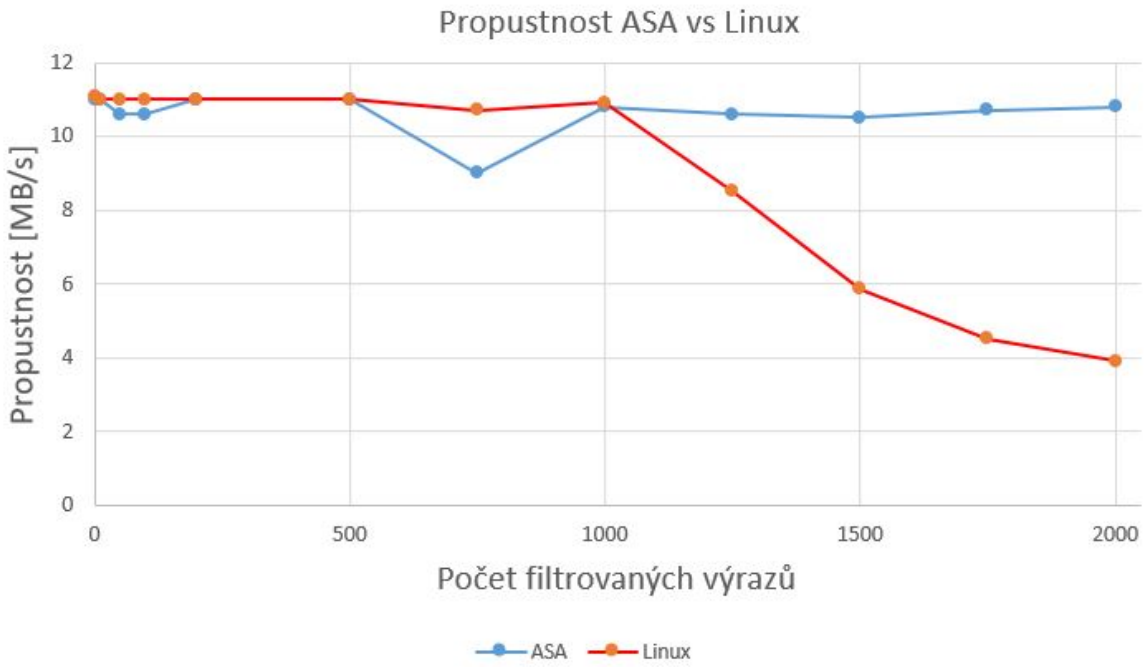
Cena Aplikační firewall na bázi ip6tables je možno implementovat na každém zařízení s Linux systémem. Pro použití v domácí síti s malými nároky je možné použít mikropočítače typu Orange R1 (2x FastEthernet, 256MB RAM, Quad-core 32-bit Cortex-A7, IEEE 802.11 b/g/n, 500 Kč) případnější výkonnější variantu Raspberry PI. Škálovat je možné až po vysoce výkonné servery, které je možno vybavit 10 Gb síťovými kartami a dosáhnout vysoké propustnosti. Cena Linux firewallu je založena pouze na ceně hardware a nepoužívá žádnou licenční politiku. Velice kvalitní Snort IPS je možno v Linuxu vytvořit zdarma pomoci několika příkazů open-source skriptu fwsnort. Při implementaci Firewallu ASA je třeba dimenzovat model firewallu vzhledem k požadavkům při jeho použití. Prodej řady 5500 byl ukončen 16.září 2013. Nahrazující řada 5500-x pro malé a střední podniky začíná modelem ASA 5506-X s propustností 300 Mbps stavové inspekce, 125 Mbps propustnost při aplikačním dohledu a IPS s přibližnou cenou 8 956 Kč. Řada končí modelem 5555-X s propustnosti stavové inspekce 2 Gbps, 1,250 Mbps propustnost při aplikačním dohledu a IPS s přibližnou cenou 475 000 Kč. Při požadavku vysoké propustnosti datacentra jsou modely řady 5585-x s nejvyšším modelem Cisco ASA 5585-X SSP-60 s propustností stavové inspekce 40 Gbps a propustnosti při aplikačním dohledu a IPS 10 Gbps s cenou 7 300 205 Kč. Cena hardware není konečná částka pro použití ASA firewallu. Funkce firewallu bývají licencovány a je možné se setkat s mnoha limity, pro jejich funkcionalitu je třeba zakoupit licenci. V případě řady 5500-x jsou funkce firewallu licencovány odděleně od funkcí modulu Firepower. Příkladem: Cena 1 roku licence funkcí IPS, AMP, URL firepower modulu pro model 5506-x je 7 596 Kč.

Relativní výpočetní náročnost Implementace obou firewallů nelze porovnávat z hlediska výpočetní náročnosti, jelikož oba implementované firewally mají jiné procesory s jiným výpočetním výkonem. Lze porovnat pouze zatížení relativně k dostupnému výkonu jednotlivých firewallů. Firewall ASA vykázal strmý nárust využití CPU při zvyšování filtrovaných výrazů viz.: Obrázek: 26, poté ale udržel zatížení na stejné úrovni nezávisle na zvyšování filtračních nároků. Linux firewall v testu projevil lineární růst zatížení s přibývajícími nároky viz.: Obrázek37,

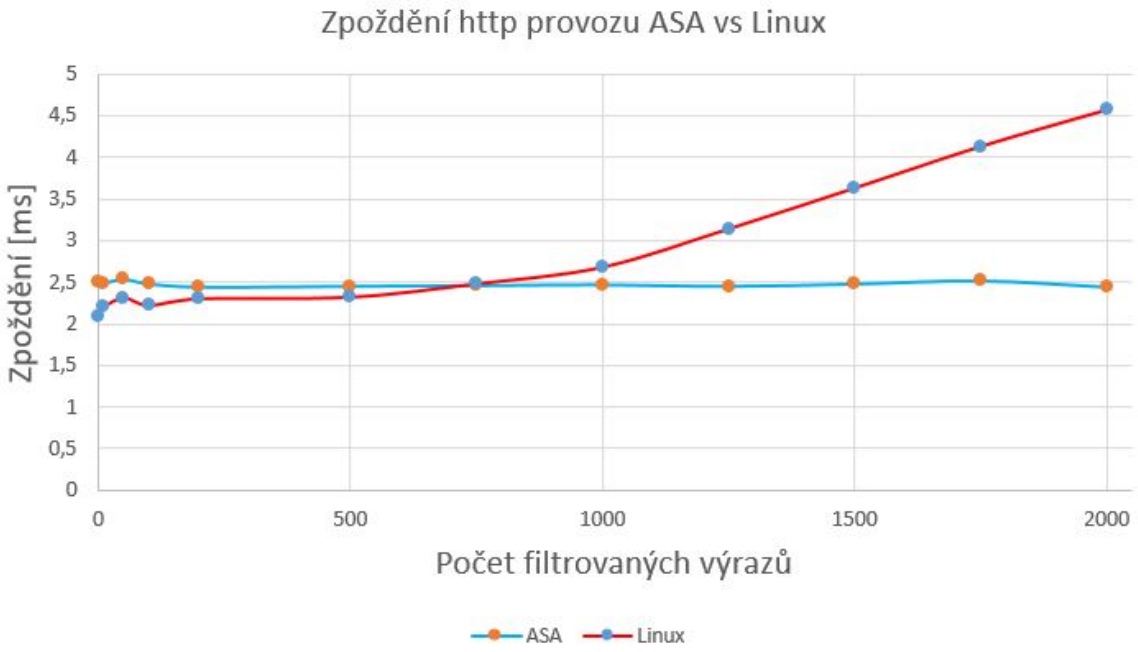
úkony filtrace orientoval více k jednomu jádru svého vícejádrového procesoru.

Vliv na provoz v síti

S dostupnými možnostmi testování nebyl prokázán žádný vliv hluboké inspekce firewallem ASA na provoz v síti. Aplikační firewall na Linuxu ovlivňuje jak propustnost tak zpoždění testovaného provozu. Porovnání vlivu obou firewallů na propustnost provozu vyobrazuje graf viz.: Obrázek: 39. Graf zpoždění provozu při zvyšování filtračních nároků na oba firewally vykresluje: Obrázek: 40. Jak je vidět v obou grafech, kritická hranice Linux firewallu byla 1000 pravidel, poté začalo zpoždění lineárně růst a zároveň klesat propustnost prohledávaného provozu.



Obrázek 39: Propustnost HTTP provozu ASA vs Linux.



Obrázek 40: Zpoždění HTTP provozu ASA vs Linux.

8 Závěr

V teoretické části, diplomová práce popisuje dnešní typy firewallů a obecné přístupy k jejich implementaci. Následně jsou popsány podrobněji principy funkce hlavních typů firewallů a možnosti, kterých lze s daným typem firewallu dosáhnout. Další teoretická část práce popsala odlišnosti protokolu IPv6 od protokolu IPv4, hierarchii adresace, typy adres a důsledky plynoucí z těchto odlišností. Poslední teoretická část věnující se protokolu IPv6 stručně popisuje protokol původně sloužící k propojení IPv6 sítí přes IPv4 internet, jehož koncept dnes funguje jako jednoduchý přechodový mechanismus připojení existující IPv4 sítě do IPv6 internetu.

V následující kapitole s přípravou laboratoře byl tento přechodový mechanismus otestován prakticky. Práce popisuje kompletní postup implementace přípojení do světa IPv6 pomocí 6to4 od registrace u poskytovatele, až po konfiguraci přechodového bodu realizováného na směrovači s operačním systémem Open WRT. Dále kapitola poskytuje postup k zprovoznění obrazu s předinstalovaným systémem s laboratorním cvičením z projektu Seed Labs, který je použit pro demonstraci útoků SQL injekce. Současně kapitola popisuje postup instalace systému Lubuntu do virtuálního prostředí Oracle Virtual Box. Následuje postup instalace a nastavení FTP serverů.

Další praktická kapitola pokračuje nastavením Cisco ASA firewallu k použití v síti s podporou protokolu IPv6. Práce poté demonstruje jednoduchost použití SQL injekce proti testovací webové stránce, je proveden aplikační rozbor infikovaných dat a následně vysvětlen postup obrany proti tomuto typu útoku jak na firewallu ASA pomocí hluboké aplikační inspekce, tak pomocí ip6tables modulu string plnícího stejnou funkci v případě firewallu Linux. Kompletní konfigurace firewallů poté obsahuje 10 blokačních výrazů navržených tak, aby blokovali také veškeré dostupné modifikace vycházejících z těchto výrazů. Tyto patří mezi výrazy nejčastěji použité k obcházení autentifikace webu a přihlášení admin uživatele bez platného hesla. Dále je testován vliv zvyšování nároků hluboké aplikační inspekce na přenosové parametry síťového provozu a jejich vliv na výpočetní prostředky obou firewallů. Testování firewallů proběhlo při lineárním zvyšování potencionálně nebezpečných výrazů o délce 20 znaků do počtu 2000. Zvyšování počtu filtrovaných výrazů u firewallu ASA nepřineslo žádné omezení přenosové rychlosti v dostupném přenosovém pásmu 100mbps. Stejný test byl proveden i pro firewall Linux. Od počtu 1000 hledaných výrazů začal firewall lineárně snižovat přenosovou rychlost. Po dosažení inspekce dvou tisíc výrazů firewall omezoval propustnost HTTP provozu z 100 mbps na pouhých 31 mbps. Firewall tedy inspekcí snížil propustnost provozu o 65%. S podobnou linearitou byl zaznamenán i nárust v odezvě jednotlivých dotazů, kde z průměrných 2,1 ms stoupla odezva na 4,57 ms. Inspekce firewallu ASA nebyla v provozu jakkoli zaznamenatelná. Dále je praktická část orientována k možnostem inspekce protokolu FTP. Pro ilustraci složitosti nastavení inspekce firewallu ASA je prováděna konfigurace pomocí příkazové řádky, kterou doplňuje modifikace konfigurace přes grafickou aplikaci ASDM. Stejná konfigurace byla implementována i na Linuxu a tudíž práce vyobrazuje kontrast složitosti jednotlivých návazností při konfiguraci ASA proti jednoduchosti konfigurace inspekce firewallem Linux.

Práce přinesla porovnání realizace aplikačního firewallu na open-source platformě proti komerčnímu proprietárnímu řešení. Vzhledem k přesně vymezeným hranicím vlastností komerčních řešení by se práce mohla dále odvíjet pouze směrem open-source. Výhodou open-source je možnost modifikace a kompilace vlastních funkcí a vylepšení v existujícím řešení. Naskytuje se tedy možnost optimalizace zpracování paketových front pro zvýšení výkonu Linux aplikačního firewallu pomocí rozdělení paketových front na více CPU.

Literatura

[1] SATRAPA, Pavel. Internetový protokol verze 6. 3. Praha: CZ.NIC, z. s. p. o, 2011. ISBN 978-80-904248-4-5.

[2] RASH, Michael. Linux firewalls: attack detection and response with iptables, psad, and fwsnort. San Francisco: No Starch Press, c2007. ISBN 978-1-59327-141-1.

[3] PROCHÁZKA, Ivo. LABORATORNÍ ÚLOHA PREZENTUJÍCÍ APLIKAČNÍ FIREWALL. Brno, 2017. Bakalářská práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Ing. Zdeněk Martinásek, Ph.D.

[4] ROESCH, Marty. Snort: Rule Doc Search [online]. [cit. 2019-04-16]. Dostupné z: `https://www.snort.org/rule_docs?utf8=%E2%9C%93&rules_query=&submit_rule_search=`

[5] IP6TABLES [online]. man2html, 2015 [cit. 2018-12-05]. Dostupné z: http://ipset.netfilter.org/ip6tables.man.html

[6] Iptables-extensions [online]. man2html, 2015 [cit. 2018-12-05]. Dostupné z: http://ipset.netfilter.org/iptables-extensions.man.html

[7] RASH, Michael. Firewall Snort [online]. [cit. 2019-04-16]. Dostupné z:`https://github.com/mrash/fwsnort`

[8] Turris: Oficiální dokumentace [online]. Praha: CZ.NIC, z. s. p. o. [cit. 2019-04-28]. Dostupné z: `https://doc.turris.cz/doc/cs/howto/vlan_settings_omnia?s[]=turris&s[]=omnia&s[]=vlan`

[9] Internet Statistics & Facts (Including Mobile) for 2018 - HostingFacts.com [online]. Dostupné z: https://hostingfacts.com/internet-facts-stats/

[10] IoT Analytics: State of the IoT 2018 [online]. 2018 [cit. 2018-10-10]. Dostupné z: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

[11] Check Point Software: Next Generation Firewall [online]. Israel: Check Point Software Technologies, 2018 [cit. 2018-10-26]. Dostupné z: https://www.checkpoint.com/products/next-generation-firewall/

[12] Technical Support & Documentation - Cisco Systems: Configure the SSL decryption on FirePOWER Module using ASDM (On-Box Management) [online]. San Jose USA: Cisco Systems, 2016 [cit. 2018-10-26]. Dostupné z: https://www.cisco.com/c/en/us/support/docs/security/asa-5500-x-firepower-services/200577-Configure-the-SSL-decryption-on-FirePOWE.html

[13] Cisco ASA with FirePOWER Services Data Sheet [online]. San Jose, California, United States: Cisco Systems, 2019, 11.1.2019 [cit. 2019-03-14]. Dostupné z: https://cisco-apps.cisco.com/c/en/us/products/collateral/security/asa-5500-series-next-generation-firewalls/datasheet-c78-733916.html

[14] Cisco ASA Series Firewall CLI Configuration Guide: Getting Started with Application Layer Protocol Inspection. Cisco ASA Series Firewall CLI Configuration Guide, 9.1 [online]. San Jose, California, USA: Cisco Systems, 2017, 24.5.2017 [cit. 2019-04-16]. Dostupnéz:https://www.cisco.com/c/en/us/td/docs/security/asa/asa91/configuration/firewall/asa_91_firewall_config/inspect_overview.html

[15] Cisco Services for IPS: End-of-Sale for Cisco Services for Intrusion Prevention System Support Program. Cisco Services for IPS: End-of-Sale for Cisco Services for Intrusion Prevention System Support Program [online]. [cit. 2019-04-16]. Dostupné z: https://tools.cisco.com/security/center/ipshome.x?i=62&shortna=CiscoIPSSignatures#CiscoIPSSignatures

[16] Sourcefire Next Generation IPS. Sourcefire Next Generation IPS [online]. San Jose, California, United States: Cisco Systems, 2019 [cit. 2019-03-14]. Dostupné z: https://www.cisco.com/c/dam/global/th_th/assets/docs/seminar/Sourcefire_Next_Generation_IPS_Datasheet.pdf

[17] Next-Generation Firewall Defined By Gartner [online]. Santa Clara: Palo Alto Networks, 2009 [cit. 2018-10-26]. Dostupné z: https://researchcenter.paloaltonetworks.com/2009/10/next-generation-firewall-defined-by-gartner/

[18] Stateful firewall. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2018 [cit. 2018-12-05]. Dostupné z: https://en.wikipedia.org/wiki/Stateful_firewall

[19] Gitbub. Bombardier [online]. San Francisco, 2016 [cit. 2019-01-06]. Dostupné z: https://github.com/codesenberg/bombardier

[20] Gitbub. Bombardier [online]. San Francisco, 2016 [cit. 2019-01-06]. Dostupné z: https://github.com/codesenberg/bombardier/releases

[21] The Go Programming Language. The Go Programming Language [online]. The Go Authors, 2018 [cit. 2019-01-06]. Dostupné z: https://golang.org/doc/install

[22] CHACON, Scott. Git. Git [online]. Git comunity, 2018 [cit. 2019-01-06]. Dostupné z: https://git-scm.com/download/win

[23] ORACLE, Virtual box [online]. [cit. 2019-01-06]. Dostupné z: https://www.virtualbox.org/wiki/Downloads

[24] DU, Wenliang. Seed Labs: Lab Environment Setup. Seed Labs: Lab Environment Setup [online]. Syracuse, New York: Syracuse University, 2018, Květen 2018 [cit. 2019-04-29]. Dostupné z: `http://www.cis.syr.edu/~wedu/seed/lab_env.html`

[25] SEED Labs: Labs for Security Education. SEED Labs: User Manual of the Pre-built Ubuntu 16.04 Virtual Machine [online]. Syracuse, New York: Syracuse University, 2018 [cit. 2019-04-29]. Dostupné z: `http://www.cis.syr.edu/~wedu/seed/Documentation/Ubuntu16_04_VM/Ubuntu16_04_VM_Manual.pdf`

[26] Lubuntu Alternate 64-bit [online]. [cit. 2019-03-10]. Dostupné z: https://lubuntu.net/downloads/

A Konfigurace laboratorního přepínače

```
Current configuration : 2367 bytes
!
version 12.1
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname LAB_SWITCH
!
!
ip subnet-zero
!
ip ssh time-out 120
ip ssh authentication-retries 3
vtp mode transparent
!
spanning-tree mode pvst
no spanning-tree optimize bpdu transmission
spanning-tree extend system-id
!
!
!
!
vlan 10
 name OUTSIDE
!
vlan 20
 name INTRA_ASA
!
vlan 22
 name INTRA_LINUX_FW
!
interface FastEthernet0/1
 description Linux Firewall INSIDE
 switchport access vlan 22
 switchport mode access
```

```
 spanning-tree portfast
!
interface FastEthernet0/2
!
interface FastEthernet0/3
!
interface FastEthernet0/4
 description ASA Firewall INSIDE
 switchport access vlan 20
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/5
!
interface FastEthernet0/6
!
interface FastEthernet0/7
 description Linux Firewall OUTSIDE
 switchport access vlan 10
 spanning-tree portfast
!
interface FastEthernet0/8
 description ASA Firewall OUTSIDE
 switchport access vlan 10
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/9
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/10
 description ASA servers access port
 switchport access vlan 20
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/11
 description LinuxFW servers access port
```

```
 switchport access vlan 22
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/12
!
interface FastEthernet0/13
 spanning-tree portfast
!
interface FastEthernet0/14
 spanning-tree portfast
!
interface FastEthernet0/15
 spanning-tree portfast
!
interface FastEthernet0/16
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/17
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/18
!
interface FastEthernet0/19
 spanning-tree portfast
!
interface FastEthernet0/20
 spanning-tree portfast
!
interface FastEthernet0/21
!
interface FastEthernet0/22
 switchport mode access
 shutdown
 spanning-tree portfast
!
interface FastEthernet0/23
```

```
 switchport mode access
 spanning-tree portfast
!
interface FastEthernet0/24
 description TURRIS DMZ
 switchport access vlan 10
 switchport mode access
 spanning-tree portfast
!
interface Vlan1
 no ip address
 no ip route-cache
 shutdown
!
ip http server
!
line con 0
line vty 0 4
 login
line vty 5 15
 login
!
!
end
```

B Celá konfigurace CISCO ASA firewallu

```
: Hardware:   ASA5506, 4096 MB RAM, CPU Atom C2000 series 1250 MHz, 1
    CPU (4 cores)
:
ASA Version 9.4(1)
!
hostname ASAv6
enable password 8Ry2YjIyt7RRXU24 encrypted
xlate per-session deny tcp any4 any4
xlate per-session deny tcp any4 any6
xlate per-session deny tcp any6 any4
xlate per-session deny tcp any6 any6
xlate per-session deny udp any4 any4 eq domain
xlate per-session deny udp any4 any6 eq domain
xlate per-session deny udp any6 any4 eq domain
xlate per-session deny udp any6 any6 eq domain
names
!
interface GigabitEthernet1/1
 shutdown
 no nameif
 no security-level
 no ip address
!
interface GigabitEthernet1/2
 shutdown
 no nameif
 no security-level
 no ip address
!
interface GigabitEthernet1/3
 shutdown
 no nameif
 no security-level
 no ip address
!
interface GigabitEthernet1/4
 nameif inside
```

```
 security-level 100
 ipv6 address 2001:470:736e:b::1/64
 ipv6 enable
 ipv6 nd managed-config-flag
!
interface GigabitEthernet1/5
 shutdown
 no nameif
 no security-level
 no ip address
!
interface GigabitEthernet1/6
 shutdown
 no nameif
 no security-level
 no ip address
!
interface GigabitEthernet1/7
 shutdown
 no nameif
 no security-level
 no ip address
!
interface GigabitEthernet1/8
 nameif outside
 security-level 0
 no ip address
 ipv6 address 2001:470:736e:a::2/64
 ipv6 enable
!
interface Management1/1
 management-only
 shutdown
 no nameif
 security-level 0
 no ip address
!
regex test "admin%27or\+[1]%3D[1]\+or\+%27%27%3D%27"
```

```
regex SQL-regex_8r "admin\x27\+or\+\x27[1]\x27\x3d\x27[1]|admin%27\+
    or\+%27[1]%27%3D%27[1]|(admin)[']%20or%20'1'='1"
regex accfile ".*\.[Aa][Cc][Cc]"
regex avifile ".*\.[Aa][Vv][Ii]"
regex SQL-regex_signID5930 "[uU][nN][iI][oO][nN](%20|\x2b|\x27)([aA][
    lL][lL](%20|\x2b|\x27))[sS][eE][lL][eE][cC][tT]"
regex SQL_regex_29r "admin"\+or\+"[1]"%3D"[1]|admin"\+or\+"[1]"\x3D"
    [1]"
regex SQL-regex_signID5474 "[Ss][Ee][Ll][Ee][Cc][Tt](%2[0bB]|+)[^\r\
    x00-\x19\x7f-\xff]+(%2[0bB]|+)[Ff][Rr][Oo][Mm](%2[0bB]|+)"
regex SQL-regex_12r "admin\x27\or\+[1]\x3D[1]\+or\+\x27\x27\x3D\x27|
    admin%27or\+[1]%3D[1]\+or\+%27%27%3D%27"
regex SQL-regex_21r "admin%27%29\+or\+%27[1]%27%3D%27[1]|admin\x27\
    x29\+or\+\x27[1]\x27\x3D\x27[1]"
regex SQL-regex_13r "admin%27\+or\+[1]%3D[1]|admin\x27\+or\+[1]\x3D
    [1]"
regex mkvfile ".*\.[Mm][Kk][Vv]"
regex SQL-regex_33r "admin"%29\+or\+%28"[1]"%3D"[1]|admin"\x29\+or\+\
    x28"[1]"\x3D"[1]"
regex wawfile ".*\.[Ww][Aa][Ww]"
regex mp3file ".*\.[Mm][Pp]3"
regex mp4file ".*\.[Mm][Pp]4"
regex SQL-regex_17r "admin%27%29\+or\+%28%27[1]%27%3D%27[1]|admin\x27
    \x29\+or\+\x28\x27[1]\x27\x3D\x27[1]"
regex SQL-regex_37r "admin"%29\+or\+"[1]"%3D"[1]|admin"\x29\+or\+"[1]
    "\x3D"[1]"
regex flacfile ".*\.[Ff][Ll][Aa][Cc]"
regex wmvfile ".*\.[Ww][Mm][Vv]"
ftp mode passive
clock timezone CEST 1
clock summer-time CEDT recurring last Sun Mar 2:00 last Sun Oct 3:00
object network obj_any
 subnet 0.0.0.0 0.0.0.0
object service tcp5901
 service tcp source range 1 65535 destination eq 5901
 description vnc
object service tcp3306
 service tcp source range 1 65535 destination eq 3306
 description sql
```

```
object-group service InGlobalServices
 service-object icmp6
 service-object tcp-udp destination eq www
 service-object tcp destination eq ftp
object-group service OutGlobalServices
 service-object ip
 service-object icmp6
object-group protocol TCPUDP
 protocol-object udp
 protocol-object tcp
access-list outside_access_in extended permit object-group
    InGlobalServices any 2001:470:736e:b::/64
access-list inside_access_in extended permit object-group
    OutGlobalServices 2001:470:736e:b::/64 any6
pager lines 24
logging enable
logging asdm-buffer-size 512
logging trap debugging
logging asdm debugging
mtu inside 1500
mtu outside 1500
no failover
no monitor-interface service-module
icmp unreachable rate-limit 1 burst-size 1
asdm image disk0:/asdm-741.bin
no asdm history enable
arp timeout 14400
no arp permit-nonconnected
access-group inside_access_in in interface inside
access-group outside_access_in in interface outside
ipv6 route outside ::/0 2001:470:736e:a::1
timeout xlate 3:00:00
timeout pat-xlate 0:00:30
timeout conn 1:00:00 half-closed 0:10:00 udp 0:02:00 icmp 0:00:02
timeout sunrpc 0:10:00 h323 0:05:00 h225 1:00:00 mgcp 0:05:00 mgcp-
    pat 0:05:00
timeout sip 0:30:00 sip_media 0:02:00 sip-invite 0:03:00 sip-
    disconnect 0:02:00
timeout sip-provisional-media 0:02:00 uauth 0:05:00 absolute
```

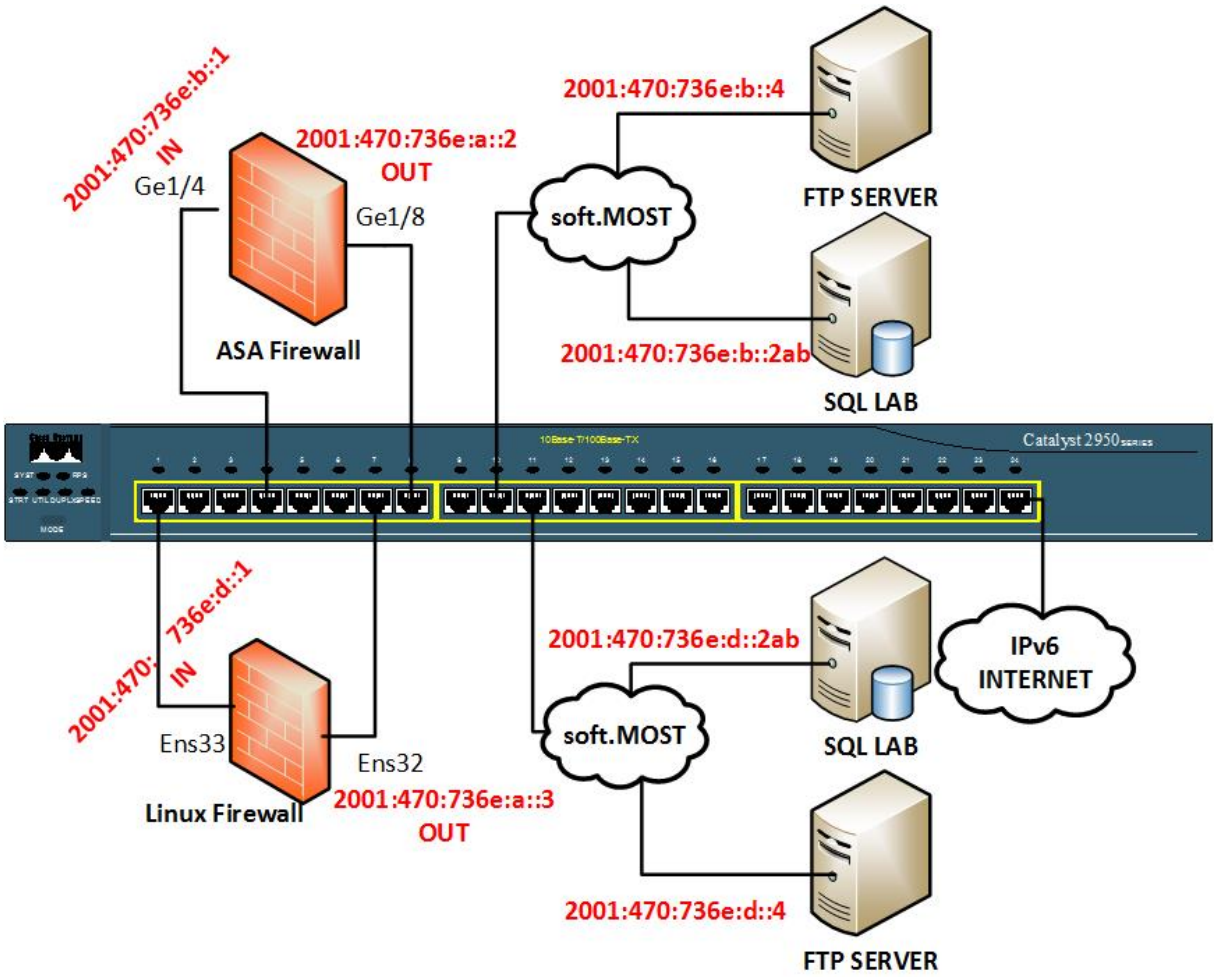
```
timeout tcp-proxy-reassembly 0:01:00
timeout floating-conn 0:00:00
user-identity default-domain LOCAL
aaa authentication http console LOCAL
aaa authentication ssh console LOCAL
http server enable
http 2001:470:736e:b::/64 inside
http 2001:718:1001::/48 outside
http 2001:470:736e:10::/64 outside
no snmp-server location
no snmp-server contact
service sw-reset-button
crypto ipsec security-association pmtu-aging infinite
crypto ca trustpoint _SmartCallHome_ServerCA
 no validation-usage
 crl configure
crypto ca trustpool policy
crypto ca certificate chain _SmartCallHome_ServerCA
telnet timeout 5
ssh stricthostkeycheck
ssh 2001:470:736e:b::/64 inside
ssh 2001:718:1001::/48 outside
ssh 2001:470:736e:10::/64 outside
ssh timeout 60
ssh version 2
ssh key-exchange group dh-group14-sha1
console timeout 0
dhcpd auto_config inside
!
threat-detection basic-threat
threat-detection statistics access-list
threat-detection statistics tcp-intercept rate-interval 30 burst-rate
    400 average-rate 200
dynamic-filter enable
dynamic-filter ambiguous-is-black
dynamic-access-policy-record DfltAccessPolicy
username uzivatel password hpSibxPh3CfRlTOu encrypted privilege 15
!
class-map FTP-class
```

```
 match port tcp eq ftp
class-map type regex match-any SQL_INJECTION
 match regex SQL-regex_13r
 match regex SQL-regex_12r
 match regex SQL-regex_17r
 match regex SQL-regex_33r
 match regex SQL-regex_21r
 match regex SQL-regex_signID5930
 match regex SQL-regex_37r
 match regex SQL-regex_signID5474
 match regex SQL_regex_29r
 match regex SQL-regex_8r
class-map SQL-class
 match port tcp eq www
class-map type regex match-any MEDIA_FILES
 match regex wawfile
 match regex flacfile
 match regex wmvfile
 match regex mp3file
 match regex mkvfile
 match regex alice
 match regex mp4file
 match regex avifile
class-map global_policy
class-map type inspect http match-all asdm_medium_security_methods
 match not request method post
 match not request method get
 match not request method head
class-map inspection_default
 match default-inspection-traffic
class-map type inspect ftp match-all BlockUpload
 match filename regex class MEDIA_FILES
 match request-command put stou
class-map type inspect http match-all asdm_high_security_methods
 match not request method get
 match not request method head
!
!
policy-map type inspect dns preset_dns_map
```

```
 parameters
  message-length maximum client auto
  message-length maximum 512
policy-map type inspect ftp FTP_INSPECT
 parameters
  mask-banner
  mask-syst-reply
 match request-command rnfr rnto
  reset log
 class BlockUpload
  reset log
policy-map type inspect ipv6 Ivp6Test
 parameters
 match header routing-type range 0 255
  drop
policy-map global_policy
 class inspection_default
  inspect dns preset_dns_map
  inspect ftp
  inspect h323 h225
  inspect h323 ras
  inspect rsh
  inspect rtsp
  inspect esmtp
  inspect sqlnet
  inspect skinny
  inspect sunrpc
  inspect xdmcp
  inspect sip
  inspect netbios
  inspect tftp
  inspect ip-options
  inspect icmp
policy-map type inspect http SQL_Inject
 parameters
  protocol-violation action reset log
 match request body regex class SQL_INJECTION
  reset log
 match request uri regex class SQL_INJECTION
```

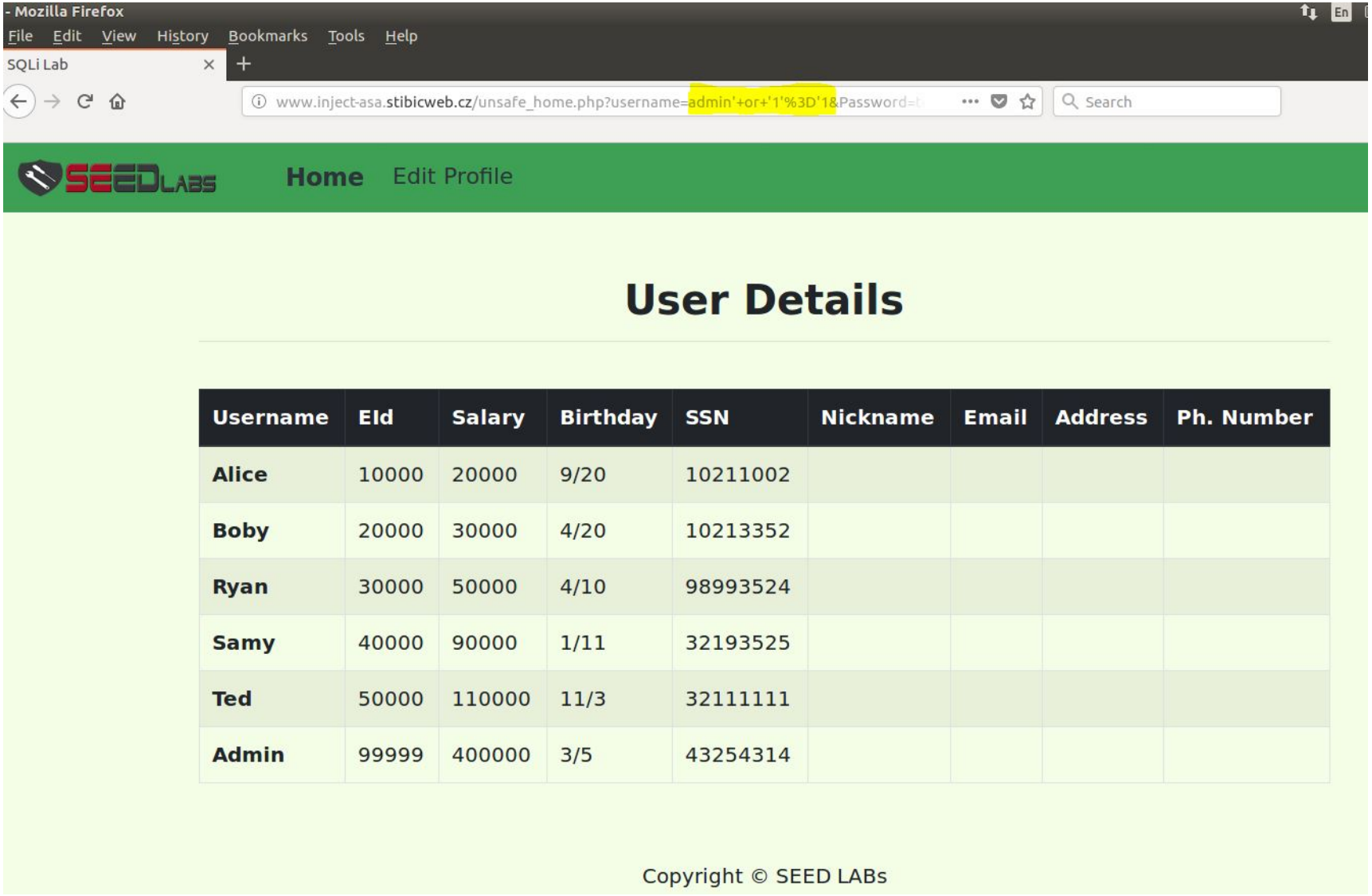
```
  reset log
policy-map outside-policy
 class SQL-class
  inspect http SQL_Inject
 class FTP-class
  inspect ftp strict FTP_INSPECT
policy-map type inspect dns migrated_dns_map_1
 parameters
  message-length maximum client auto
  message-length maximum 512
!
service-policy global_policy global
service-policy outside-policy interface outside
prompt hostname context
call-home reporting anonymous
call-home
 profile CiscoTAC-1
  no active
  destination address http https://tools.cisco.com/its/service/oddce/
   services/DDCEService
  destination address email callhome@cisco.com
  destination transport-method http
  subscribe-to-alert-group diagnostic
  subscribe-to-alert-group environment
  subscribe-to-alert-group inventory periodic monthly 25
  subscribe-to-alert-group configuration periodic monthly 25
  subscribe-to-alert-group telemetry periodic daily
hpm topN enable
Cryptochecksum:fd5f5d1e64fbf74816cfeb9e3fba0154
: end
```

C Schéma zapojení laboratoře



Obrázek 41: Topologie laboratorní sítě.

D Úspěšné přihlášení administrátora s použitím SQL injekce



Obrázek 42: Demonstrace SQL injekce.

E ASA skript pro regulární výrazy

```
#!/bin/bash
counter=1

while [ $counter -le 2000 ]
do
NEW_STRING=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 15 |
    head -n 1)

echo regex bigregex$counter '"'$NEW_STRING'"' >> createregex.txt
echo no regex bigregex$counter >> deleteregex.txt
echo match regex bigregex$counter >> addmatchregex.txt
echo no match regex bigregex$counter >> removematchregex.txt
((counter++))
done
```

F Sript pro počítání paketů za sekundu

```
#!/bin/bash

time="1"     # čas v sekundách
int="enp0s3"   # měřené rohraní
while true
    do
        txpkts_old="`cat /sys/class/net/$int/statistics/tx_packets`"
   # odeslané pakety
        rxpkts_old="`cat /sys/class/net/$int/statistics/tx_packets`"
   # přijaté pakety
            sleep $time
        txpkts_new="`cat /sys/class/net/$int/statistics/tx_packets`"
   # odeslané pakety
                rxpkts_new="`cat /sys/class/net/$int/statistics/
   tx_packets`" # přijaté pakety
        txpkts="`expr $txpkts_new - $txpkts_old`"            # výpoč
   et odeslaných paketů v čase
        rxpkts="`expr $rxpkts_new - $rxpkts_old`"            # výpoč
   et přijatých paketů v čase
            echo "tx $txpkts pkts/s - rx $rxpkts pkts/ on interface
   $int"
    done
```

G Tabulky měření Firewallů

pravidla	pakety/s	cpu1[%]	dotaz/s	zpoždění[ms]	rychlost[MB/s]
1	814	5	400	2,51	0,91
10	810	8	409	2,49	0,92
50	814	16	392	2,54	0,91
100	812	18	402	2,48	0,91
200	806	19	410	2,44	0,91
500	814	19	407	2,45	0,91
750	809	19	406	2,46	0,91
1000	801	19	408	2,47	0,91
1250	810	18	408	2,45	0,91
1500	812	18	412	2,48	0,91
1750	798	18	396	2,52	0,89
2000	816	19	409	2,44	0,91

Tabulka 6: Měřené parametry ASA FW, test bombardier.

pravidla	pakety/s	cpu1[%]	cpu2[%]	systém CPU[%]
1	960	2,7	2	1,5
10	900	3	2	2
50	900	2,7	2	2
100	900	4	2	2
200	900	4,3	2	2,2
500	900	5,5	2	2
750	880	7,5	2,3	3,33
1000	780	9	3,3	4
1250	670	11	4,8	4,76
1500	560	14	4,6	5
1750	500	14,8	4,8	6
2000	450	18	6	7

Tabulka 7: Měřené parametry Linux FW(1/2), test bombardier.

pravidla	dotaz/s	zpoždění[ms]	rychlost[MB/s]
1	456	2,10	1,05
10	454	2,20	1,02
50	432	2,31	0,97
100	448	2,22	1,1
200	434	2,30	0,98
500	430	2,32	0,97
750	402	2,48	0,91
1000	373	2,68	0,86
1250	317	3,14	0,73
1500	275	3,63	0,63
1750	242	4,13	0,55
2000	218	4,57	0,5

Tabulka 8: Měřené parametry Linux FW(2/2), test bombardier.

pravidla	**paket/s**	**CPU1**[%]	**CPU2**[%]	**Systém CPU**[%]	**rychlost**[MB/s]
1	8300	3,00	3	3	11,1
10	8300	5	6	5	11
50	8300	6	5	5	11
100	8300	7	7	5	11
200	8200	7	9	6	11
500	8100	22	22	21	11
750	8100	32	28	25	10,7
1000	8100	55	35	40	10,9
1250	6300	73	25	46	8,5
1500	4500	83	25	40	5,86
1750	3500	88	7	34	4,51
2000	3100	92	5	40	3,9

Tabulka 9: Měřené parametry Linux FW, test stahování.

H Celá konfigurace blokace SQL injekce pro Linuxu

```
sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27+or+%271%27%3D%27" --algo bm -j LOG --log-prefix "SQL-
  injection8"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27+or+%271%27%3D%27" --algo bm -j REJECT --reject-with tcp-
  reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27or+1%3D1+or+%27%27%3D%27" --algo bm -j LOG --log-prefix "
  SQL-injection12"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27or+1%3D1+or+%27%27%3D%27" --algo bm -j REJECT --reject-
  with tcp-reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27+or+1%3D1" --algo bm -j LOG --log-prefix "SQL-injection13"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27+or+1%3D1" --algo bm -j REJECT --reject-with tcp-reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27%29+or+%28%271%27%3D%271" --algo bm -j LOG --log-prefix "
  SQL-injection17"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27%29+or+%28%271%27%3D%271" --algo bm -j REJECT --reject-
  with tcp-reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27%29+or+%271%27%3D%271" --algo bm -j LOG --log-prefix "SQL-
```

```
  injection21"

sudo ip6tables -A FORWARD -p tcp --dport 80  -m string --string "
  admin%27%29+or+%271%27%3D%271" --algo bm -j REJECT --reject-with
  tcp-reset


sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  admin%22+or+%221%22%3D%221" --algo bm -j LOG --log-prefix "SQL-
  injection29"

sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  admin%22+or+%221%22%3D%221" --algo bm -j REJECT --reject-with tcp-
  reset


sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  admin%22or+1%3D1+or+%22%22%3D%22" --algo bm -j LOG --log-prefix "
  SQL-injection33"

sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  admin%22or+1%3D1+or+%22%22%3D%22" --algo bm -j REJECT --reject-
  with tcp-reset


sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  admin%22%29+or+%28%221%22%3D%221" --algo bm -j LOG --log-prefix "
  SQL-injection37"

sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  admin%22%29+or+%28%221%22%3D%221" --algo bm -j REJECT --reject-
  with tcp-reset


sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  UNION%2BSELECT" --algo bm -j LOG --log-prefix "SQL-ID5930"

sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
  UNION%2BSELECT" --algo bm -j REJECT --reject-with tcp-reset
```

```
sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "%2
  B%29SELECT%2B*%2BFROM%2B" --algo bm -j LOG --log-prefix "SQL-
  ID5930"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "%2B
  %29SELECT%2B*%2BFROM%2B" --algo bm -j REJECT --reject-with tcp-
  reset
```

I Skript pro generování náhodných pravidel pro Linux

```
#!/bin/bash
target = 2000 // parametr pro cílový počet pravidel
counter=1
sudo ip6tables -F
while [ $counter -le $target ]
do
NEW_STRING=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 20 |
    head -n 1)

sudo ip6tables -A FORWARD -p tcp --dport 80  -m string --string
   $NEW_STRING --algo bm -j REJECT --reject-with tcp-reset
((counter++))
done

 sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
   admin%27+or+%271%27%3D%27" --algo bm -j LOG --log-prefix "SQL-
   injection8"

 sudo ip6tables -A FORWARD  -p tcp --dport 80 -m string --string "
   admin%27+or+%271%27%3D%27" --algo bm -j REJECT --reject-with tcp-
   reset


 sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
   admin%27or+1%3D1+or+%27%27%3D%27" --algo bm -j LOG --log-prefix "
   SQL-injection12"

 sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
   admin%27or+1%3D1+or+%27%27%3D%27" --algo bm -j REJECT --reject-
   with tcp-reset


 sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
   admin%27+or+1%3D1" --algo bm -j LOG --log-prefix "SQL-injection13"

 sudo ip6tables -A FORWARD  -p tcp --dport 80  -m string --string "
   admin%27+or+1%3D1" --algo bm -j REJECT --reject-with tcp-reset
```

```
sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27%29+or+%28%271%27%3D%271" --algo bm -j LOG --log-prefix "
  SQL-injection17"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27%29+or+%28%271%27%3D%271" --algo bm -j REJECT --reject-
  with tcp-reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27%29+or+%271%27%3D%271" --algo bm -j LOG --log-prefix "SQL-
  injection21"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%27%29+or+%271%27%3D%271" --algo bm -j REJECT --reject-with
  tcp-reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%22+or+%221%22%3D%221" --algo bm -j LOG --log-prefix "SQL-
  injection29"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%22+or+%221%22%3D%221" --algo bm -j REJECT --reject-with tcp-
  reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%22or+1%3D1+or+%22%22%3D%22" --algo bm -j LOG --log-prefix "
  SQL-injection33"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%22or+1%3D1+or+%22%22%3D%22" --algo bm -j REJECT --reject-
  with tcp-reset
```

```
sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%22%29+or+%28%221%22%3D%221" --algo bm -j LOG --log-prefix "
  SQL-injection37"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  admin%22%29+or+%28%221%22%3D%221" --algo bm -j REJECT --reject-
  with tcp-reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  UNION%2BSELECT" --algo bm -j LOG --log-prefix "SQL-ID5930"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "
  UNION%2BSELECT" --algo bm -j REJECT --reject-with tcp-reset


sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "%2
  B%29SELECT%2B*%2BFROM%2B" --algo bm -j LOG --log-prefix "SQL-
  ID5930"

sudo ip6tables -A FORWARD -p tcp --dport 80 -m string --string "%2B
  %29SELECT%2B*%2BFROM%2B" --algo bm -j REJECT --reject-with tcp-
  reset

echo All done
sudo ip6tables -L
```

J Celá konfigurace FTP firewallu pro Linux

```
sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mp3" --algo bm -j LOG --log-
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mp3" --algo bm -j REJECT

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mp4" --algo bm -j LOG --log-
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mp4" --algo bm -j REJECT

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".avi" --algo bm -j LOG --log-
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".avi" --algo bm -j REJECT

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".acc" --algo bm -j LOG --log-
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".acc" --algo bm -j REJECT

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".waw" --algo bm -j LOG --log-
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".waw" --algo bm -j REJECT

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".flac" --algo bm -j LOG --log-
```

```
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".flac" --algo bm -j REJECT

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".wmv" --algo bm -j LOG --log-
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".wmv" --algo bm -j REJECT

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mkv" --algo bm -j LOG --log-
    prefix "MediaBlock"

sudo ip6tables -A FORWARD -p tcp --dport 21 -m state --state
    ESTABLISHED -m string --string ".mkv" --algo bm -j REJECT
```